Genstat®

Directives

www.vsni.co.uk

VSNi

Genstat® Reference Manual (Release 22)

Part 2: Directives

Genstat Release 22 was developed by VSN International Ltd, in collaboration with practising statisticians at Rothamsted and other organisations in Britain, Australia, New Zealand and The Netherlands.

Published by: VSN International, 2 Amberside, Wood Lane, Hemel Hempstead, Hertfordshire HP2 4TP, UK

E-mail: info@genstat.co.uk

Website: http://www.genstat.co.uk/

First published 1996, as the *Genstat 5 Release 3.2 Command Language Manual*
This edition published 2022, for Genstat Release 22

Citation: VSN International (2022). *Genstat Reference Manual (Release 22), Part 2 Directives*. VSN International, Hemel Hempstead, UK.

Contents

Conventions

Genstat system words are shown in the Courier typeface e.g. `CALCULATE`. In the general form of each statement, elements of the language to be substituted by the user are in italics, e.g. *variate*. New directives in Release 22, or options and parameters of existing directives that have been modified in Release 22, are marked by the symbol †.

List of directives in Release 22

ADD adds extra terms to a linear, generalized linear, generalized additive, or nonlinear model.
ADDPOINTS adds points for new objects to a principal coordinates analysis.
ADISPLAY displays further output from analyses produced by ANOVA.
AFMINABERRATION forms minimum aberration factorial or fractional-factorial designs.
AFRESPONSESURFACE uses the BLKL algorithm to construct designs for estimating response surfaces.
AGRCRESOLVABLE forms doubly resolvable row-column designs.
AKEEP copies information from an ANOVA analysis into Genstat data structures.
ANOVA analyses y-variates by analysis of variance according to the model defined by earlier BLOCKSTRUCTURE, COVARIATE, and TREATMENTSTRUCTURE statements.
ASRULES derives association rules from transaction data.
ASSIGN sets elements of pointers and dummies.
AXES defines the axes in each window for high-resolution graphics.
AXIS defines an oblique axis for high-resolution graphics.
BARCHART plots bar charts in high-resolution graphics.
BASSESS assesses potential splits for regression and classification trees.
BCUT cuts a tree at a defined node, discarding nodes and information below it.
BGROW adds new branches to a node of a tree.
BIDENTIFY identifies specimens using a tree.
BJOIN extends a tree by joining another tree to a terminal node.
BLOCKSTRUCTURE defines the blocking structure of the design and hence the strata and the error terms.
BREAK suspends execution of the statements in the current channel or control structure and takes subsequent statements from the channel specified.
CALCULATE calculates numerical values for data structures.
CALLS lists library procedures called by a procedure.
CAPTION prints captions in standardized formats.
CASE introduces a "multiple-selection" control structure.
CATALOGUE displays the contents of a backing-store file.
CLOSE closes files.
CLUSTER forms a non-hierarchical classification.
COKRIGE calculates kriged estimates using a model fitted to the sample variograms and cross-variograms of a set of variates.
COLOUR defines the red, green and blue intensities to be used for the Genstat colours with certain graphics devices.
COMBINE combines or omits "slices" of a multi-way data structure (table, matrix, or variate).
COMMANDINFORMATION provides information about whether (and how) a command has been implemented.
CONCATENATE concatenates and truncates lines (units) of text structures; allows the case of letters to be changed.
CONTOUR is a synonym for LPCONTOUR.
COPY forms a transcript of a job.
CORRELATE forms correlations between variates, autocorrelations of variates, and lagged cross-correlations between variates.
COVARIATE specifies covariates for use in subsequent ANOVA statements.
CVA performs canonical variates analysis.
DBITMAP plots a bit map of RGB colours.
DCLEAR clears a graphics screen.
DCONTOUR draws contour plots on a plotter or graphics monitor.

DDISPLAY redraws the current graphical display.
DEBUG puts an implicit BREAK statement after the current statement and after every NSTATEMENTS subsequent statements, until an ENDDEBUG is reached.
DECLARE declares one or more customized data structures.
DELETE deletes the attributes and values of structures.
DEVICE switches between (high-resolution) graphics devices.
DFINISH ends a sequence of related high-resolution plots.
DGRAPH draws graphs on a plotter or graphics monitor.
DHISTOGRAM draws histograms on a plotter or graphics monitor.
DIAGONALMATRIX declares one or more diagonal matrix data structures.
DISPLAY prints, or reprints, diagnostic messages.
DISTRIBUTION estimates the parameters of continuous and discrete distributions.
DKEEP saves information from the last plot on a particular device.
DLOAD loads the graphics environment settings from an external file.
DPIE draws a pie chart on a plotter or graphics monitor.
DREAD reads the locations of points from an interactive graphical device.
DROP drops terms from a linear, generalized linear, generalized additive, or nonlinear model.
DSAVE saves the current graphics environment settings to an external file.
DSHADE plots a shade diagram of 3-dimensional data.
DSTART starts a sequence of related high-resolution plots.
DSURFACE produces perspective views of a two-way arrays of numbers.
DUMMY declares one or more dummy data structures.
DUMP prints information about data structures, and internal system information.
DUPLICATE forms new data structures with attributes taken from an existing structure.
D3GRAPH plots a 3-dimensional graph.
D3HISTOGRAM plots three-dimensional histograms.
EDIT edits text vectors.
ELSE introduces the default set of statements in block-if or in multiple-selection control structures.
ELSIF introduces a set of alternative statements in a block-if control structure.
ENDBREAK returns to the original channel or control structure and continues execution.
ENDCASE indicates the end of a "multiple-selection" control structure.
ENDDEBUG cancels a DEBUG statement.
ENDFOR indicates the end of the contents of a loop.
ENDIF indicates the end of a block-if control structure.
ENDJOB ends a Genstat job.
ENDPROCEDURE indicates the end of the contents of a Genstat procedure.
ENQUIRE provides details about files opened by Genstat.
EQUATE transfers data between structures of different sizes or types (but the same modes i.e. numerical or text) or where transfer is not from single structure to single structure.
ESTIMATE is a synonym for TFIT.
EXECUTE executes the statements contained within a text.
EXIT exits from a control structure.
EXPRESSION declares one or more expression data structures.
EXTERNAL declares an external function in a DLL for use by the OWN function.
FACROTATE rotates factor loadings from a principal components, canonical variates or factor analysis.
FACTOR declares one or more factor data structures.
FARGUMENTS forms lists of arguments involved in an expression.
FAULT checks whether to issue a diagnostic, i.e. a fault, warning or message.
FCA performs factor analysis.

FCLASSIFICATION forms a classification set for each term in a formula, breaks a formula up into separate formulae (one for each term), and applies a limit to the number of factors and variates in the terms of a formula.
FCOPY makes copies of files.
FCOVARIOGRAM forms a covariogram structure containing auto-variograms of individual variates and cross-variograms for pairs from a list of variates.
FDELETE deletes files.
FILTER is a synonym for TFILTER.
FIT fits a linear, generalized linear, generalized additive, or generalized nonlinear model.
FITCURVE fits a standard nonlinear regression model.
FITNONLINEAR fits a nonlinear regression model or optimizes a scalar function.
FKEY forms design keys for multi-stratum experimental designs, allowing for confounded and aliased treatments.
FLRV forms the values of LRV structures.
FOR introduces a loop; subsequent statements define the contents of the loop, which is terminated by the directive ENDFOR.
FORECAST is a synonym for TFORECAST.
FORMULA declares one or more formula data structures.
FOURIER calculates cosine or Fourier transforms of real or complex series.
FPSEUDOFACTORS determines patterns of confounding and aliasing from design keys, and extends the treatment model to incorporate the necessary pseudo-factors.
FRAME defines the positions and appearamce of the plotting windows within the frame of a high-resolution graph.
FRENAME renames files.
FRQUANTILES forms regression quantiles.
FSIMILARITY forms a similarity matrix or a between-group-elements similarity matrix or prints a similarity matrix.
FSSPM forms the values of SSPM structures.
FTSM forms preliminary estimates of parameters in time-series models.
FVARIOGRAM forms experimental variograms.
GENERATE generates factor values for designed experiments.
GET accesses details of the "environment" of a Genstat job.
GETATTRIBUTE accesses attributes of structures.
GETLOCATIONS finds locations of an identifier within a pointer, or a string within a factor or text, or a number within any numerical data structure.
GRAPH is a synonym for LPGRAPH.
GROUPS forms a factor (or grouping variable) from a variate or text, together with the set of distinct values that occur.
HCLUSTER performs hierarchical cluster analysis.
HDISPLAY displays results ancillary to hierarchical cluster analyses: matrix of mean similarities between and within groups, a set of nearest neighbours for each unit, a minimum spanning tree, and the most typical elements from each group.
HELP provides help information about Genstat.
HISTOGRAM is a synonym for LPHISTOGRAM.
HLIST lists the data matrix in abbreviated form.
HREDUCE forms a reduced similarity matrix (referring to the GROUPS instead of the original units).
HSUMMARIZE forms and prints a group by levels table for each test together with appropriate summary statistics for each group.
IF introduces a block-if control structure.
INPUT specifies the input file from which to take further statements.

INTERPOLATE interpolates values at intermediate points.
IRREDUNDANT forms irredundant test sets for the efficient identification of a set of objects.
JOB starts a Genstat job.
KRIGE calculates kriged estimates using a model fitted to the sample variogram.
LIST lists details of the data structures currently available within Genstat.
LPCONTOUR produces contour maps of two-way arrays of numbers using character (i.e. line-printer) graphics.
LPGRAPH produces point and line plots using character (i.e. line-printer) graphics.
LPHISTOGRAM produces histograms using character (i.e. line-printer) graphics.
LRV declares one or more LRV data structures.
MARGIN forms and calculates marginal values for tables.
MATRIX declares one or more matrix data structures.
MCOVARIOGRAM fits models to sets of variograms and cross-variograms.
MDS performs non-metric multidimensional scaling.
MERGE copies subfiles from backing-store files into a single file.
MODEL defines the response variate(s) and the type of model to be fitted for linear, generalized linear, generalized additive, and nonlinear models.
MONOTONIC fits an increasing monotonic regression of y on x.
NAG calls an algorithm from the NAG Library.
NNDISPLAY displays output from a multi-layer perceptron neural network fitted by NNFIT.
NNFIT fits a multi-layer perceptron neural network.
NNPREDICT forms predictions from a multi-layer perceptron neural network fitted by NNFIT.
OPEN opens files.
OPTION defines the options of a Genstat procedure with information to allow them to be checked when the procedure is executed.
OR introduces a set of alternative statements in a "multiple-selection" control structure.
OUTPUT defines where output is to be stored or displayed.
OWN does work specified in Fortran subprograms linked into Genstat by the user.
PAGE moves to the top of the next page of an output file.
PARAMETER defines the parameters of a Genstat procedure with information to allow them to be checked when the procedure is executed.
PASS does work specified in subprograms supplied by the user, but not linked into Genstat. This directive may not be available on some computers.
PCO performs principal coordinates analysis, also principal components and canonical variates analysis (but with different weighting from that used in CVA) as special cases.
PCORELATE relates the observed values on a set of variables to the results of a principal coordinates analysis.
PCP performs principal components analysis.
PEN defines the properties of "pens" for high-resolution graphics.
POINTER declares one or more pointer data structures.
PREDICT forms predictions from a linear or generalized linear model.
PRINT prints data in tabular format in an output file, unformatted file, or text.
PROCEDURE introduces a Genstat procedure.
QDIALOG produces a modal dialog box to obtain a response from the user.
QRD calculates QR decompositions of matrices.
RANDOMIZE randomizes the units of a designed experiment or the elements of a factor or variate.
RBDISPLAY displays output from a radial basis function model fitted by RBFIT.
RBFIT fits a radial basis function model.
RBPREDICT forms predictions from a radial basis function model fitted by RBFIT.
RCYCLE controls iterative fitting of generalized linear, generalized additive, and nonlinear

models, and specifies parameters, bounds etc for nonlinear models.
RDISPLAY displays the fit of a linear, generalized linear, generalized additive, or nonlinear model.
READ reads data from an input file, an unformatted file, or a text.
RECORD dumps a job so that it can later be restarted by a RESUME statement.
REDUCE is a synonym for HREDUCE.
REFORMULATE modifies a formula or an expression to operate on a different set of data structures.
RELATE is a synonym for PCORELATE.
REML fits a variance-components model by residual (or restricted) maximum likelihood.
RENAME assigns new identifiers to data structures.
RESTRICT defines a restricted set of units of vectors for subsequent statements.
RESUME restarts a recorded job.
RETRIEVE retrieves structures from a subfile.
RETURN returns to a previous input stream (text vector or input channel).
RFUNCTION estimates functions of parameters of a nonlinear model.
RKEEP stores results from a linear, generalized linear, generalized additive, or nonlinear model.
RKESTIMATES saves estimates and other information about individual terms in a regression analysis.
ROTATE does a Procrustes rotation of one configuration of points to fit another.
SCALAR declares one or more scalar data structures.
SET sets details of the "environment" of a Genstat job.
SETALLOCATIONS runs through all ways of allocating a set of objects to subsets.
SETCALCULATE performs Boolean set calculations on the contents of vectors or pointers.
SETOPTION sets or modifies defaults of options of Genstat directives or procedures.
SETPARAMETER sets or modifies defaults of parameters of Genstat directives or procedures.
SETRELATE compares two sets of values in two data structures.
SET2FORMULA forms a model formula using structures supplied in a pointer.
SHELLEXECUTE launches executables or opens files in another application using their file extension.
SKIP skips lines in input or output files.
SORT sorts units of vectors according to an index vector.
SPLOAD loads Genstat spreadsheet files.
SSPM declares one or more SSPM data structures.
STEP selects terms to include in or exclude from a linear, generalized linear, or generalized additive model according to the ratio of residual mean squares.
STOP ends a Genstat program.
STORE to store structures in a subfile of a backing-store file.
STRUCTURE defines a compound data structure.
SUSPEND suspends execution of Genstat to carry out commands in the operating system. This directive may not be available on some computers.
SVD calculates singular value decompositions of matrices.
SWITCH adds terms to, or drops them from a linear, generalized linear, generalized additive, or nonlinear model.
SYMMETRICMATRIX declares one or more symmetric matrix data structures.
SYNTAX obtains details of the syntax of a command and the source code of a procedure.
TABLE declares one or more table data structures.
TABULATE forms summary tables of variate values.
TDISPLAY displays further output after an analysis by TFIT.
TERMS specifies a maximal model, containing all terms to be used in subsequent linear,

generalized linear, generalized additive, and nonlinear models.
TEXT declares one or more text data structures.
TFILTER filters time series by time-series models.
TFIT estimates parameters in Box-Jenkins models for time series.
TFORECAST forecasts future values of a time series.
TKEEP saves results after an analysis by TFIT.
TRANSFERFUNCTION specifies input series and transfer-function models for subsequent estimation of a model for an output series.
TREATMENTSTRUCTURE specifies the treatment terms to be fitted by subsequent ANOVA statements.
TREE declares a tree, & initializes it to have a single node known as the root.
TRY displays results of single-term changes to a linear, generalized linear, or generalized additive model.
TSM declares one or more TSM data structures.
TSUMMARIZE displays characteristics of time series models.
TXBREAK breaks up a text structure into individual words.
TXCONSTRUCT forms a text structure by appending or concatenating values of scalars, variates, texts, factors, pointers or formulae; allows the case of letters to be changed or values to be truncated and reversed.
TXFIND finds a subtext within a text structure.
TXINTEGERCODES converts textual characters to and from their corresponding integer codes.
TXPOSITION locates strings within the lines of a text structure.
TXREPLACE replaces a subtext within a text structure.
TX2VARIATE converts text structures to variates.
UNITS defines an auxiliary vector of labels and/or the length of any vector whose length is not defined when a statement needing it is executed.
VARIATE declares one or more variate data structures.
VCOMPONENTS defines the variance-components model for REML.
VCYCLE controls details of the REML algorithm.
VDISPLAY displays further output from a REML analysis.
VKEEP copies information from a REML analysis into Genstat data structures.
VPEDIGREE generates an inverse relationship matrix for use when fitting animal or plant breeding models by REML.
VPREDICT forms predictions from a REML model.
VRESIDUAL defines the residual term for a REML model.
VSTATUS prints the current model settings for REML.
VSTRUCTURE defines a variance structure for random effects in a REML model
WORKSPACE accesses private data structures for use in procedures.
XAXIS defines the x-axis in each window for high-resolution graphics.
YAXIS defines the y-axis in each window for high-resolution graphics.
ZAXIS defines the z-axis in each window for high-resolution graphics.
%CD changes the current directory.
%CLOSE closes the binary file opened by %OPEN.
%FPOSITION returns the current position in the binary file opened by %OPEN.
%LOG adds text into the Input Log window in the Genstat client.
%MESSAGEBOX displays text in a dialog in the Genstat client.
%OPEN open a binary file for use with %WRITE.
%SLEEP pauses execution of the server for a time specified in seconds.
%TEMPFILE creates a unique temporary file in the Genstat temporary folder.
%WRITE writes values of data structures to a binary file opened by %OPEN.

ADD

Adds extra terms to a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate, unchanged); default unch
CONSTANT = *string token*	How to treat the constant (estimate, omit, unchanged, ignore); default unch
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
AOVDESCRIPTION = *text*	Description for line in accumulated analysis of variance (or deviance) table when POOL=yes

Parameter

formula	List of explanatory variates and factors, or model formula

Description

ADD adds terms to the current regression model, which may be linear, generalized linear, generalized additive, standard curve or nonlinear. It is best to give a TERMS statement before investigating sequences of models using ADD, in order to define a common set of units for the models that are to be explored. If no model has been fitted since the TERMS statement, the current model is taken to be the null model.

The model fitted by ADD will include a constant term if the previous model included one, and

will not include one if the previous model did not. You can, however, change this using the CONSTANT option.

The options of ADD are almost all the same as those of the FIT directive, and are described there. There is also an extra option NONLINEAR. This is relevant when fitting curves. For example, if we have a variate Dilution and a factor Solution, the program below will fit parallel curves for the different solutions.

```
MODEL Density
TERMS Dilution * Solution
FITCURVE [PRINT=model,estimates; CURVE=logistic] \
  Dilution + Solution
```

If we then put

```
ADD Dilution.Solution
```

the curves are still constrained to have common nonlinear parameters, but all linear parameters are estimated separately for each group. Alternatively, if we put

```
ADD [NONLINEAR=separate] Dilution.Solution
```

different nonlinear parameters will be estimated for each solution too; so only the information about variability is pooled.

Options: PRINT, NONLINEAR, CONSTANT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION, PROBABILITY, AOVDESCRIPTION.
Parameter: unnamed.

Action with RESTRICT

If a TERMS statement was given before fitting the model, any restrictions on the variates or factors in the model will have been implemented then. So any restrictions on vectors involved in the model specified by ADD will be ignored. If no TERMS statement has been given and ADD introduces new terms into the model, restrictions on the variates or factors in these terms will be taken into account and may cause the units involved in the regression to be redefined.

See also

Directives: MODEL, TERMS, FIT, FITCURVE, DROP, SWITCH, TRY.
Functions: COMPARISON, POL, REG, LOESS, SSPLINE.
Genstat Reference Manual 1 Summary section on: Regression analysis.

ADDPOINTS

Adds points for new objects to a principal coordinates analysis.

Option

PRINT = *string tokens*	Printed output required (coordinates, residuals); default * i.e. no printing

Parameters

NEWDISTANCES = *matrices*	Squared distances of the new objects from the original points
LRV = *LRVs*	Latent roots and vectors from the PCO analysis
CENTROID = *diagonal matrices*	Centroid distances from the PCO analysis
COORDINATES = *matrices*	Saves the coordinates of the additional points in the space of the original points
RESIDUALS = *matrices* or *variates*	Saves the residuals of the new objects from that space

Description

The input to ADDPOINTS is specified by the first three parameters. The NEWDISTANCES parameter specifies an $s \times n$ matrix containing squared distances of the s new units from the n old units. The LRV and CENTROID parameters specify structures defining the configuration of old units; these have usually been produced by a PCO statement.

The PRINT option controls the printed output with settings:

coordinates	to print the coordinates of the new points;
residuals	to print the residual distances of the new units from the coordinates in the space of the old units.

The other parameters can be used to save the results: the COORDINATES parameter allows you to specify an $s \times k$ matrix to save the coordinates for the new units, and the residuals can be saved in an $s \times 1$ matrix using the RESIDUALS parameter. The value k is determined by the dimensionality of the input coordinates from the preceding PCO statement.

Option: PRINT.
Parameters: NEWDISTANCES, LRV, CENTROID, COORDINATES, RESIDUALS.

See also

Directives: PCO, PCORELATE.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

ADISPLAY

Displays further output from analyses produced by ANOVA.

Options

PRINT = *string tokens*	Output from the analyses of the y-variates, adjusted for any covariates (aovtable, information, covariates, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default * i.e. no printing
UPRINT = *string tokens*	Output from the unadjusted analyses of the y-variates (aovtable, information, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default * i.e. no printing
CPRINT = *string tokens*	Output from the analyses of the covariates, if any (aovtable, information, effects, residuals, contrasts, means, %cv, missingvalues); default * i.e. no printing
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
PFACTORIAL = *scalar*	Limit on number of factors in printed tables of means or effects; default 9
PCONTRASTS = *scalar*	Limit on order of printed contrasts; default 9
PDEVIATIONS = *scalar*	Limit on number of factors in a treatment term whose deviations from the fitted contrasts are to be printed; default 9
FPROBABILITY = *string token*	Printing of probabilities for variance ratios in the aov table (yes, no); default no
PSE = *string tokens*	Standard errors to be printed with tables of means, PSE=* requests s.e.'s to be omitted (differences, lsd, means); default diff
TWOLEVEL = *string token*	Representation of effects in 2^n experiments (responses, Yates, effects); default resp
NOMESSAGE = *string tokens*	Which warning messages to suppress (nonorthogonal, residual); default *
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5

Parameter

identifiers	Save structure (from ANOVA) to provide details of each analysis from which information is to be displayed; if omitted, output is from the most recent ANOVA

Description

The ADISPLAY directive allows you to display further output from one or more analyses of variance, without having to repeat all the calculations. You can store the information from each analysis in a save structure, using ANOVA, and then specify the same structure in the SAVE parameter of ADISPLAY. Several save structures can be listed, corresponding to the analyses of several different variates. They need not all have been produced by the same ANOVA statement nor even be from the same design. Alternatively, if you just want to display output from the last y-variate that was analysed, you need not specify the SAVE parameter in either ANOVA or

ADISPLAY: the save structure for the last y-variate analysed is saved automatically, and provides the default for ADISPLAY.

Apart from CHANNEL, all the options of ADISPLAY also occur with ANOVA and are described there. CHANNEL can be set to a scalar to divert the output to another output channel. Alternatively, it can specify the identifier of text data structure to store the output (and in fact an undeclared structure will be defined as a text, automatically).

The other difference concerns the options for printed output. The default for PRINT with ADISPLAY is different from that with ANOVA. You are most likely to use ADISPLAY when you are working interactively, to examine one component of output at a time, and it is not obvious that any one component will then be more popular than any other. So the default for ADISPLAY produces no output (that is, PRINT=*). This also means that you do not need to suppress the output explicitly when you are using UPRINT and CPRINT to examine components of output from analysis of covariance. Also, the settings information, covariates, and missingvalues have a slightly different effect with ANOVA than with ADISPLAY. As they are part of the default specified for ANOVA, they will not produce any output unless there is something definite to report. With ADISPLAY you need to request them explicitly, so Genstat will always produce some sort of report. For example, putting

```
ADISPLAY [PRINT=missing]
```

when there are no missing values will simply tell you there are none.

Options: PRINT, UPRINT, CPRINT, CHANNEL, PFACTORIAL, PCONTRASTS, PDEVIATIONS, FPROBABILITY, PSE, TWOLEVEL, NOMESSAGE, LSDLEVEL.
Parameter: unnamed.

See also

Directives: ANOVA, BLOCKSTRUCTURE, COVARIATE, TREATMENTSTRUCTURE, AKEEP.
Procedures: AGRAPH, APLOT, AMCOMPARISON, AMDUNNETT, APOLYNOMIAL, ARESULTSUMMARY, AUDISPLAY, A2DISPLAY.
Genstat Reference Manual 1 Summary section on: Analysis of variance.

AFMINABERRATION

Forms minimum aberration factorial or fractional-factorial designs.

Options

PRINT = *string tokens*	Controls printed output (summary, keyblocks, keydefining, monitoring); default *
NTIMES = *scalar*	Number of designs to try in a random search; default 0 does the full search
SEED = *scalar*	Seed for the random number generator used to search the designs randomly; default 0

Parameters

LEVELS = *scalars*	Number of levels of the treatment factors, must be a power of a prime number
NTREATMENTFACTORS = *scalars*	Number of treatment factors
NUNITS = *scalars*	Number of units in each block of a block design or in the principal block of a fractional factorial
NSUBUNITS = *scalars*	Number of units in each (sub-)block
KEYBLOCKS = *matrices*	Design key for the blocks and sub-blocks
KEYDEFINING = *matrices*	Design key specifying the defining contrasts
RESOLUTION = *scalars*	Saves the resolution of the design
ABERRATION = *scalars*	Saves the aberration of the design
SUBRESOLUTION = *scalars*	Saves the resolution of the sub-design
SUBABERRATION = *scalars*	Saves the aberration of the sub-design
NDESIGN = *scalars*	Saves or defines the design number
NSUBDESIGN = *scalars*	Saves or defines the sub-design number

Description

The concept of *minimum aberration* provides an effective way of selecting either a full factorial design where treatment contrasts are confounded with blocks, or a fractional factorial. (Essentially, these are equivalent – the fractional factorial design is formed by taking only one block of the full factorial.) The *resolution* of the design is defined as the largest integer *r* such that no interaction term with *r* factors is confounded with blocks (or aliased). The *aberration* of the design is the number of interaction terms with *r*+1 factors that are confounded (or aliased). A *minimum aberration* design is a design with the smallest aberration out of the designs with the highest available resolution. It is thus a design that is closest to the next level of resolution.

AFMINABERRATION searches for minimum aberration designs using the algorithm of Laycock & Rowley (1995), and we gratefully acknowledge Patrick Laycock's assistance with the implementation into Genstat. The number of treatment factors is specified by the NFACTORS parameter. Their number of levels is specified by the LEVELS parameter. This must be an integer power of a prime number. The number of units in each block (or the number of plots in the equivalent fractional factorial) is specified by the NUNITS parameter, and must be a power of LEVELS.

AFMINABERRATION can also form a sub-blocking factor that can be used to define blocks if the design is to be used to form a fractional factorial. The number of units in each sub-block is defined by the NSUBBLOCKS parameter (and again must be a power of LEVELS).

If there are very many designs to search, you may prefer to examine only a random selection. The NTIMES option sets the number of designs to try; its default of zero requests the standard (full) search. The SEED option sets the seed for the random numbers that are used to select the designs randomly; the default of zero continues the existing sequence or (if none) initializes the seed automatically. (Note that this version of the random number generator is shared with other

design construction algorithms, such as FKEY.)

Printed output is controlled by the PRINT option, with settings:

summary	summarizes the design properties;
keyblocks	prints a design key to generate the block and sub-block factors from the treatment factor (or pseudo-factors to generate them if they have more than *p* levels);
keydefining	prints a design key specifying the defining contrasts i.e. all the treatment contrasts confounded with blocks or sub-blocks;
monitoring	prints monitoring information about the design construction.

You can save the design keys using the KEYBLOCKS and KEYDEFINING parameters. In addition, the NDESIGN parameter can save a unique "design number" for the design, and the NSUBDESIGN parameter can save a unique number for the sub-design of the design. You can input these with NDESIGN and NSUBDESIGN later, along with the same settings for NTREATMENTFACTORS, LEVELS, NUNITS and NSUBUNITS, to obtain the design keys without repeating the design search. The RESOLUTION and ABERRATION parameters can save the resolution and aberration of the (main) design, and the SUBRESOLUTION and SUBABERRATION parameters can save the resolution and aberration of a sub-design.

Options: PRINT, NTIMES, SEED.

Parameters: LEVELS, NTREATMENTFACTORS, NUNITS, NSUBUNITS, KEYBLOCKS, KEYDEFINING, RESOLUTION, ABERRATION, SUBRESOLUTION, SUBABERRATION, NDESIGN, NSUBDESIGN.

Reference

Laycock, P.J. & Rowley, P.J. (1995). A method for generating and labelling all regular fractions or blocks for q^{n-m} designs. *Journal of the Royal Statistical Society, Series B*, **57**, 191-204.

See also

Directives: AFRESPONSESURFACE, FKEY, FPSEUDOFACTORS.

Procedures: AGFACTORIAL, AKEY, ARANDOMIZE, ASAMPLESIZE, FACPRODUCT.

Genstat Reference Manual 1 Summary sections on: Design of experiments, Analysis of variance.

AFRESPONSESURFACE

Uses the BLKL algorithm to construct designs for estimating response surfaces.

Options

PRINT = *string token*	Printed output required (monitoring); default * i.e. no printing
TERMS = *formula*	Model to be fitted when the design is used; no default i.e. this option must be specified
CONSTANT = *string token*	How to treat the constant in the model (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of terms in the model; default 2
NUNITS = *scalar*	Number of units (or trials) in the design
NDELETION = *scalar*	Number of design points to consider for deletion; default takes NUNITS/4, or 4 is this is larger
NINCLUSION = *scalar*	Number of design points to consider for inclusion; default takes NUNITS/4, or 4 is this is larger
NRUNS = *scalar*	Number of times to run the algorithm; default 100
ADJUSTMENTSTEP = *scalar*	Maximum amount by which to perturb the design points in the adjustment algorithm; default * i.e. no adjustments are tried
NBLOCKS = *scalar*	Number of blocks; default 1 i.e. design not blocked
BLOCKFACTOR = *factor*	Saves the block factor (if any) for the design
BLOCKSIZE = *scalar* or *variate*	Number of units in each block of the design
PREVIOUSBLOCKS = *factor*	Supplies values of the blocking factor for any previous experiments that are to be included in the analysis of the results of the design
MIXTURE = *variates*	Lists any variates that are part of a mixture (their values must be greater than zero and sum to one)
SEED = *scalar*	Seed for random numbers used to construct the initial design; default 124195
DETERMINANT = *scalar*	Saves the determinant of the information matrix for the best design
MEANGRID = *scalar*	Saves the mean value of the standardized variance of predictions obtained from the design over a grid of x-values
MAXGRID = *scalar*	Saves the maximum value of the standardized variance of predictions obtained from the design over a grid of x-values
NGRIDPOINTS = *scalar*	Number of grid points in each x-direction to use for MEANGRID and MAXGRID; default 5

Parameters

X = *variates*	Lists the variates to be investigated in the design; these need not be supplied if none of the other parameters are required
X2 = *variates*	Lists identifiers to be used to represent squares of the x-variates in the model
X3 = *variates*	Lists identifiers to be used to represent cubes of the x-variates in the model
SUPPORTPOINTS = *variates*	Support points for each x-variate in the design; if these are not (all) specified, they are formed automatically

PREVIOUSVALUES = *variates*	Supplies values of the x-variates for any previous experiments that are to be included in the analysis of the results of the design

Description

AFRESPONSESURFACE uses the BLKL algorithm of Atkinson & Donev (1992) to construct a design to estimate parameters of a response-surface model. The algorithm searches for a D-optimal design: that is, a design that will provide a maximum value for the determinant of the information matrix of the model parameters. The model is specified using the TERMS option, with the CONSTANT option indicating whether or not it is to contain the constant term (or intercept). The FACTORIAL sets a limit on the number of variates in each model term; by default this is 2.

The NUNITS option specifies the number of units in the design. If there is to be a blocking factor in the design, the NBLOCKS option specifies its number of levels, and the BLOCKFACTOR option saves its values. The BLOCKSIZE option specifies the number of units to be contained in each block of the design, in a scalar (if they are all the same) or a variate. If the block sizes are fixed, the specified sizes must sum to the number of units. However, if you specify sizes that sum to a value greater that the required number of units, the algorithm will search for the optimum block sizes.

When the model is to contain squares or cubes of x-variables, you will need to specify identifiers to represent these using the parameters of the directive. (When using regression directives such as FIT to fit the model, you can use the POL function but this is not recognised by AFRESPONSESURFACE.) The x-variates in the model must then all be listed by the X parameter. The corresponding squares are listed by the X2 parameter, and the cubes by the X3 parameter.

After specifying the X parameter, you can also use the SUPPORTPOINTS parameter to specify the x-values of the points to be considered when constructing the design; if this is not specified, these support points are formed automatically. Note that the variates are all assumed to be scaled to have values between −1 and 1. However, the criterion for D-optimality is unaffected by linear transformations of the X-variables. So you can rescale afterwards in any way you like. AFRESPONSESURFACE allows for a set of mixture variates, whose values must all be positive and which must sum to 1. The variates in the mixture are specified using the MIXTURE option.

The PREVIOUSVALUES parameter can be used to supply values of the x-variates for any previous experiments that are to be included in the analysis of the results of the new experiment, or to specify points that must be included in the design. The PREVIOUSBLOCKS option should then indicate the blocks to which these previous observations belonged.

The BLKL algorithm starts by forming an initial design by making a random selection of points from the set of support points. The SEED option defines the seed for the random numbers used to make the selection (default 124195). The algorithm then uses an exchanges algorithm to improve the design. At each exchange, the *K* points with the lowest variance of prediction amongst the points of design are considered for replacement by the *L* points with the highest variance of prediction amongst the candidate points for inclusion in the design. The algorithm makes the best one of these exchanges, continuing until there are none that increase the determinant. The values for *K* and *L* are specified by the NDELETION and NINCLUSION options respectively. The best values depend on the design parameters, including the number of model parameters and the number of residual degrees of freedom. If they are unset, AFRESPONSESURFACE sets them to the number of units divided by 4, or 4 if this larger. The NRUNS option can be set to request that the algorithm is run several times, with different starting designs; the default is 100. The design parameters are saved only for the best design found, but you can set option PRINT=monitoring to print information about each attempt.

There is also be a final adjustment algorithm which can be used except when the design

contains mixtures. This examines the design points one at a time to see whether the design can be improved by moving it a small amount along any x-axis. If an increase is possible, the point providing the greatest increase is moved. The process is then repeated until no improvment is possible. This phase is selected by setting the ADJUSTMENTSTEP option to the maximum amount (e.g. 0.2) by which the point may be moved on any axis.

The DETERMINANT option allows you to saves the determinant of the information matrix for the best design. An alternative way of evaluating the design is to examine the standardized variance of the predictions that would be obtained from the design at other points, not in the design. The MEANGRID option can save the mean value of the standardized variance of prediction over a grid of x-values, and the MAXGRID option can save the maximum value. Number of grid points in each x-direction is specified by the NGRIDPOINTSMETHOD option (default 5).

Options: PRINT, TERMS, CONSTANT, FACTORIAL, NUNITS, NDELETION, NINCLUSION, NRUNS, ADJUSTMENTSTEP, NBLOCKS, BLOCKFACTOR, BLOCKSIZE, PREVIOUSBLOCKS, MIXTURE, SEED, DETERMINANT, MEANGRID, MAXGRID, NGRIDPOINTS.
Parameters: X, X2, X3, SUPPORTPOINTS, PREVIOUSVALUES.

Method

The algorithm is described in Sections 15.6 and 15.7 of Atkinson & Donev (1992). The source code of the algorithm was provided by Alex Donev. This, and his assistance generally with this Genstat implementation, is gratefully acknowledged.

Action with RESTRICT

Restrictions on the X, X2, X3, SUPPORTPOINTS or PREVIOUSVALUES parameters are ignored.

Reference

Atkinson, A.C. & Donev, A.N. (1992) *Optimum Experimental Designs*. Oxford University Press, Oxford.

See also

Procedures: AFNONLINEAR, AGBOXBEHNKEN, AGCENTRALCOMPOSITE, AGFACTORIAL, AGMAINEFFECT, RQUADRATIC.
Genstat Reference Manual 1 Summary sections on: Design of experiments, Regression analysis.

AGRCRESOLVABLE

Forms doubly resolvable row-column designs.

Options

PLOTORDER = *string token*	Defines the order in which the plots are formed into replicates (colserpentine, colbycol, rowserpentine, rowbyrow); default rowb
TIME = *scalar*	Time in seconds to spend searching for an optimal design; default 60
SEED = *scalar*	Seed for the randomization; default 0
MAXITERATIONS = *scalar*	The number of random designs to search for an optimal design; default 10000

Parameters

NROWS = *scalars*	Number of rows in the design
NCOLUMNS = *scalars*	Number of columns in the design
LEVELS = *scalar*, *variate* or *text*	Defines the number of levels or labels of the TREATMENT factor for each design
TREATMENTS = *factors*	Saves the treatment allocation in each design
ROWREPLICATES = *factors*	Saves the row replicates in each design
COLREPLICATES = *factors*	Saves the column replicates in each design
ROWS = *factors*	Saves the row locations of the plots in each design
COLUMNS = *factors*	Saves the column locations of the plots in each design
EXIT = *scalars*	Saves the exit code from the design search program (0 for success, greater than 0 for failure)

Description

AGRCRESOLVABLE creates approximately optimal row-column designs. They are formed into replicates in both the row and column directions so that they are doubly resolvable, i.e. resolvable in both row and column directions. The layout of plots must be a complete rectangular array, and the treatments must be equally replicated. This requires that the number of rows multiplied by the number of columns in the array must be equal to the number of treatments multiplied by the number of replicates. The row replicates are comprised of units in adjacent rows, and the column replicates are comprised of units in adjacent columns. This design can be thought of as a generalization of a Latin square, with each treatment occurring once in each row and column replicate.

An example design with four replicates of five treatments in a five-row by four-column array is shown below. As the number of treatments is the same as the number of rows, the column replicates are the same as the columns, so each treatment occurs once in each column. The row replicates are shaded in different colours and consist of five plots from adjacent columns. This is an optimal design, as the treatments in the five rows form a balanced incomplete block design (within the rows, each treatment occurs three times with every other treatment).

1	**5**	**4**	**3**
2	**1**	**5**	**4**
3	**2**	**1**	**5**
4	**3**	**2**	**1**
5	**4**	**3**	**2**

The number of rows and columns in the design must be specified by the NROWS and NCOLUMNS parameters, respectively. The number of treatments is specified by the LEVELS parameter, either as a scalar (defining the number explicitly), or as a variate giving the levels for the treatments, or by a text defining a name for each treatment.

The algorithm has the following constraints. There must be no more than 8000 plots in the design. There must be no more than 4000 rows, columns or treatments. There must be no more than 20 replicates. There must be at least two rows, columns and treatments, and at least three replicates. The number of columns must not be greater than the number of treatments. The number of rows can be greater than the number of treatments, but it must be a multiple of the number of treatments (and multiple replicates are then stacked in the columns).

The PLOTORDER option defines the order in which the plots are numbered:

colserpentine	column-by-column in a serpentine way e.g. top-to-bottom, and then bottom-to-top;
colbycol	column-by-column taking the same direction for every column;
rowserpentine	row-by-row in a serpentine way e.g. left-to-right, and then right-to-left;
rowbyrow	row-by-row taking the same direction for every row (default).

The TIME option specifies the maximum time in seconds to spend searching for an optimal design; default 60. For large designs, TIME should be increased. For example, 1000 seconds is recommended for more than 100 treatments, and 4000 seconds for more than 200 treatments.

The SEED parameter specifies the starting seed for the randomization process; the default of zero initializes the seed automatically.

The MAXITERATIONS option sets the maximum number of random starting designs to use in the search; default 10000. The search stops when either the TIME or the MAXITERATIONS limit is reached.

The factors for the resulting design can be saved by the TREATMENTS, ROWREPLICATES, COLREPLICATES, ROWS and COLUMNS parameters.

The EXIT parameter can save a scalar which is set to 0 if the design search has found a valid design, 1 if the design limits have been exceeded, 2 if a design is not possible, 3 if no design has been found, and 9 if not enough memory could be allocated for the design search.

Options: PLOTORDER, TIME, SEED, MAXITERATIONS.
Parameters: NROWS, NCOLUMNS, LEVELS, TREATMENTS, ROWREPLICATES, COLREPLICATES, ROWS, COLUMNS, EXIT.

Method

The treatments are allocated in a random order constrained to meet the resolvability criterion, and then the MS criterion (Shah 1960) is optimized by an exchange algorithm. This is repeated until an optimal design is found or the time limit or maximum number of starting designs is reached. The design chosen is one that minimizes the MS criterion.

Reference

Shah, K.R. (1960). Optimality criteria for incomplete block designs. *Annals of Mathematical Statistics*, **22**, 235-247.

See also

Procedure: AFRCRESOLVABLE.

Genstat Reference Manual 1 Summary section on: Design of experiments.

AKEEP

Copies information from an ANOVA analysis into Genstat data structures.

Options

FACTORIAL = *scalar*	Limit on number of factors in a model term; default 3
STRATUM = *formula*	Model term of the lowest stratum to be searched for effects; default * implies the lowest stratum
SUPPRESSHIGHER = *string token*	Whether to suppress the searching of higher strata if a term is not found in STRATUM (yes, no); default no
TWOLEVEL = *string token*	Representation of effects in 2^n experiments (responses, Yates, effects); default resp
RESIDUALS = *variate*	Saves residuals from the final stratum (as in the RESIDUALS parameter of ANOVA)
FITTEDVALUES = *variate*	Saves fitted values (data values or missing value estimates, minus the residuals from the final stratum – as in the FITTEDVALUES parameter of ANOVA)
CBRESIDUALS = *variate*	Saves the sum of the residuals from all the strata
CBCREGRESSION = *variate*	Saves the estimates of the covariate regression coefficients, combining information from all the strata
CBCVCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix of the combined estimates of the covariate regression coefficients
TREATMENTSTRUCTURE = *formula structure*	Saves the treatment formula used for the analysis
BLOCKSTRUCTURE = *formula structure*	Saves the block formula used for the analysis
AFACTORIAL = *scalar*	Saves the setting of the FACTORIAL option used in the ANOVA command that performed the analysis
WEIGHTS = *variate*	Saves the weights used in the analysis
YVARIATE = *dummy*	Dummy to be set to the y-variate of the analysis
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
AOVTABLE = *pointer*	Saves the analysis-of-variance table as a pointer with a variate or text for each column (source, d.f., s.s., m.s. etc)
EQFACTORS = *factors*	Factors whose levels are to be assumed to be equal within the comparisons between means calculated for SEMEANS
RMETHOD = *string token*	Type of residuals to form if the RESIDUALS option or parameter is set (simple, standardized); default simp
EXIT = *scalar*	Saves an exit code indicating the properties of the design
SAVE = *identifier*	Defines the Save structure (from ANOVA) that provides details of the analysis; default * gives that from the most recent ANOVA

Parameters

TERMS = *formula*	Model terms for which information is required
MEANS = *tables*	Table to store means for each term (available for treatment terms only)
SEMEANS = *tables*	Table of effective standard errors for the means, usable

	for calculating standard errors for differences between means in the table, at equal levels of the factors specified by the EQFACTORS option
SEDMEANS = *symmetric matrices*	Standard errors for comparisons between every pair of entries in the table of means
VCMEANS = *symmetric matrices*	Variances and covariances of means
EFFECTS = *tables* or *scalars*	Table or scalar (for terms with 1 d.f. when TWOLEVEL=responses or Yates) to store effects (for treatment terms only)
PARTIALEFFECTS = *tables*	Table or scalar (for terms with 1 d.f. when TWOLEVEL=responses or Yates) to store partial effects (for treatment terms only)
REPLICATIONS = *tables* or *scalars*	Table to store replications or scalar if they are all equal
RESIDUALS = *tables*	Table to store residuals (for block terms only)
DF = *scalars*	Number of degrees of freedom for each term
LSDMEANS = *symmetric matrices*	Least significant differences of means
DFMEANS = *symmetric matrices*	Degrees of freedom for comparisons between every pair of entries in the table of means
SS = *scalars*	Sum of squares for each term
EFFICIENCY = *scalars*	Efficiency factor for each term
VARIANCE = *scalars*	Unit variance for the effects of each term
RTERM = *formula structures*	Residual terms: for a treatment term this saves the lowest stratum where the term is estimated (down to the stratum specified by the STRATUM option); for a block term it saves all the strata to which it would be appropriate to compare the term
CEFFICIENCY = *scalars*	Covariance efficiency factor for each term
CREGRESSION = *variates*	Estimated regression coefficients for the covariates in the specified stratum
CVCOVARIANCE = *symmetric matrix*	Variance-covariance matrix of the covariate regression coefficients in the specified stratum
CSSP = *symmetric matrices*	Covariate sums of squares and products in the specified stratum
CONTRASTS = *pointers*	Estimates for the fitted contrasts of each treatment term, stored in a pointer to scalars or tables; units of the pointer are labelled by the contrast name (as used in the analysis-of-variance table)
XCONTRASTS = *pointers*	X-variates used to fit contrasts, as orthogonalized by ANOVA, stored in a pointer to tables; units of the pointer are labelled as for CONTRASTS
SECONTRASTS = *pointers*	Standard errors for estimated contrasts, stored in a pointer to scalars or tables; units of the pointer are labelled as for CONTRASTS
DFCONTRASTS = *pointers*	Degrees of freedom for estimated contrasts, stored in a pointer to scalars; units of the pointer are labelled as for CONTRASTS
CBMEANS = *tables*	Table to store estimates of the means, combining information from all the strata (for treatment terms only)
SECBMEANS = *tables*	Table of standard errors for the combined means, usable for calculating standard errors for differences between

	means in the table, at equal levels of the factors specified by the EQFACTORS option
SEDCBMEANS = *symmetric matrices*	Standard errors for comparisons between every pair of entries in the table of combined means
VCCBMEANS = *symmetric matrices*	Variances and covariances of combined means
LSDCBMEANS = *symmetric matrices*	Least significant differences of combined means
DFCBMEANS = *symmetric matrices*	Effective degrees of freedom for comparisons between every pair of entries in the table of combined means
CBEFFECTS = *tables* or *scalars*	Table or scalar (for terms with 1 d.f. when TWOLEVEL=responses or Yates) to store estimates of the effects, combining information from all the strata (for treatment terms only)
CBVARIANCE = *scalars*	Unit variance for the combined estimates of the effects of each term
DFCEFFECTS = *scalars*	Effective degrees of freedom for the combined estimates of the effects of each term
CBCEFFICIENCY = *scalars*	Covariance efficiency factor for the combined estimates of each term
STRATUMVARIANCE = *scalars*	Estimates of the stratum variances (for block terms only)
COMPONENT = *scalars*	Stratum variance components (for block terms only)
STATUS = *scalars*	Status code describing how the term is estimated (together with its marginal terms, if the term is a treatment term)

Description

AKEEP allows you to copy components of the output from an analysis of variance into standard Genstat data structures. You can save the information from the analysis in a save structure, using the SAVE option of ANOVA and then specify the same structure in the SAVE option of AKEEP. Alternatively, Genstat automatically stores the save structure from the last y-variate that has been analysed, and this is used as a default by AKEEP if you do not specify a save structure explicitly.

Several options are provided to save information about the analysis as a whole. The RESIDUALS and FITTEDVALUES options allow variates to be specified to store the residuals and fitted values, respectively. The residuals, like those saved by the RESIDUALS parameter of ANOVA, are taken only from the final stratum. The RMETHOD option controls whether these are simple residuals (like those printed by ANOVA – the default) or whether they are standardized according to their variances. As an alternative, the CBRESIDUALS option saves residuals that incorporate the variability from all the strata. With an orthogonal design, these are simply the sum of the residuals from every stratum. For a non-orthogonal design, they are the data values minus the combined estimates of the treatment effects. Likewise, the CBCREGRESSION option allows you to save estimates of covariate regression coefficients that combine information from all the strata, and the CBCVCOVARIANCE option can save their variances and covariances. (The estimates and their variances and covariances from each individual stratum can be saved using the CREGRESSION and CVCOVARIANCE parameters, as described below.) The AOVTABLE option saves the analysis-of-variance table, as a pointer with a variate or a text for each column of the table. The pointer elements are labelled with the column labels of the table, and the variates contain missing values where the table has blanks. These can be printed as blanks by setting option MISSING=' ' in the PRINT directive.

The TREATMENTSTRUCTURE, BLOCKSTRUCTURE and WEIGHTS options can save the treatment and block formulae, and the weights variate (if any) that were used to specify the analysis. The AFACTORIAL option can save the value used for the FACTORIAL option in the ANOVA comamnd

that did the analysis, and the YVARIATE option can be set to a dummy to point to the variate that was analysed (i.e. the variate defined by the Y parameter of ANOVA). The EXIT option can save an exit code summarizing the properties of the design; see the description of ANOVA for details.

The parameters of AKEEP save information about particular model terms in the analysis. With the TERMS parameter you specify a model formula, which Genstat expands to form the series of model terms about which you wish to save information. As in ANOVA, the FACTORIAL option sets a limit on the number of factors in each term. Any term containing more than that limit is deleted. The subsequent parameters allow you to specify identifiers of data structures to store various components of information for each of the terms that you have specified. If there are components that are not required for some of the terms, you should insert a missing identifier (*) at that point of the list. For example

```
AKEEP Source + Amount + Source.Amount; MEANS=*,*,Meangain;\
  SS=Ssource,Samount,Ssbya; VARIANCE=Vsource,*,*
```

sets up a table Meangain containing the source by amount table of means; it forms scalars Ssource, Samount and Ssbya to hold the sums of squares for Source, Amount and Source.Amount respectively, and scalar Vsource to store the unit variance for the effects of Source.

The structures to hold the information are defined automatically, so you need not declare them in advance. If you have declared any of the tables already, its classification set will be redefined, if necessary, to match the factors in the table that you wish to store. Thus Meangain here would be redefined to be classified by the factors Source and Amount, if it had previously been declared with some other set of classifying factors. Sizes of variates and symmetric matrices will also be redefined if necessary.

Many of the components are stored in tables, classified by the factors in the model term. Tables of means and effects are relevant only for treatment terms. Standard errors for a table of means can be saved using the SEMEANS parameter. For some designs, such as split-plots, different standard errors are needed for the means according to which pair of means is to be compared. The EQFACTORS option allows you to specify factors within the tables of means whose levels are assumed to be equal for the two means. Alternatively, the SEDMEANS parameter can save a symmetric matrix containing a standard error of difference for each pair of means, the VCMEANS parameter can save a symmetric matrix with the variances and covariances for the means, and the LSDMEANS parameter can save a symmetric matrix containing least significant differences. The LSDLEVEL option specifies the significance level to use; default 5(%). The DFMEANS parameter saves a symmetric matrix with the degrees of freedom for comparing each pair of means. The rows and columns of these matrices are labelled by the factor name and level (or label if available) of the mean concerned.

Tables of partial effects (saveable only for treatment terms, using the PARTIALEFFECTS parameter) differ from the usual effects, presented by Genstat, only when there is non-orthogonality. The usual effects of a treatment term are estimated after eliminating the terms that precede it in the model, whereas the partial effects are those that would be estimated after eliminating the subsequent treatment terms as well. The TWOLEVEL option controls what it stored for terms whose factors all have only two levels. The settings response (the default) or Yates generate a scalar response; whereas TWOLEVELS=effects produces a table of effects. Replications are stored in tables if the values are unequal. For equal replications you can supply either a scalar or a table, but if the saving structure has not been declared AKEEP will define it as a scalar. Tables of residuals are available only for block terms, and the RMETHOD option controls whether or not they are standardized.

Sums of squares, numbers of degrees of freedom, efficiency factors and unit variances are saved in scalars. The unit variance of a treatment term is the residual mean square of the stratum where the term is estimated, divided by its efficiency factor and covariance efficiency factor. Thus you can calculate the estimated variance of any of the effects of the term by dividing its

unit variance by the replication of the effect.

For a treatment term, the RTERM parameter can be used to save a formula containing the model term corresponding to the lowest stratum in which it is estimated (down to and including any stratum defined by the STRATUM option). This can then be used as the setting of the TERMS parameter of a subsequent AKEEP statement to obtain further information about the stratum, for example its number of residual degrees of freedom. For a block term, RTERM saves all the strata to which it would be appropriate to compare the term. So, with a block structure of

```
Blocks/Plots/Subplots
```

the command

```
AKEEP Blocks + Blocks.Plots; RTERM=Rb,Rbp
```

would define Rb as the formula !f(Blocks.Plots), and Rbp as the formula !f(Blocks.Plots,Subplots). Alternatively, with a block structure of

```
Reps/(Rows*Columns)
```

the command

```
AKEEP Reps; RTERM=Rr
```

would define Rr as the formula !f(Reps.Rows + Reps.Blocks).

There are three parameters for saving information about the covariates. To save the regression coefficients estimated in a particular stratum, you should specify the model term of the stratum with the TERMS parameter and a variate with the CREGRESSION parameter. Genstat defines the variate to have a length equal to the number of covariates, and stores the estimated regression coefficients of the covariates in the order in which they were listed in the COVARIATE statement. The CVCOVARIANCE parameter saves the variances and covariances of the estimated covariate regression coefficients, in a symmetric matrix. The CSSP parameter allows you to obtain sums of squares and products between the covariates for the specified model term. These are arranged in a symmetric matrix. The value in row *i* on the diagonal is the sum of squares for the term in the analysis of variance that has as its y-variate the *i*th covariate listed in the COVARIATE statement. The value in row *i* and column *j* is the cross-product between the effects estimated for the term in the analysis of variance of covariate *i* and those estimated for the same term in the analysis of covariate *j*.

The CONTRASTS, XCONTRASTS, SECONTRASTS and DFCONTRASTS parameters save information about contrasts. For each treatment term there will generally be several contrasts, so the information is stored in pointers with one element for each contrast. The elements are laballed by the name of the contrasts as it appears, for example, in the analysis-of-variance table.

The CBMEANS, CBSEMEANS, CBSEDMEANS, VCCBMEANS, LSDCBMEANS, DFCBMEANS, CBEFFECTS, CBVARIANCE, DFCEFFECTS, CBCEFFICIENCY and STRATUMVARIANCES parameters save details of estimates that combine information from all the strata of the design, and the COMPONENT parameter saves the stratum variance components.

In designs where there is partial confounding, and treatment terms are estimated in more than one stratum, options STRATUM and SUPPRESSHIGHER allow you to specify the strata from which the information is to be taken. This is relevant to tables of effects and partial effects, sums of squares, efficiency factors, unit variances, sums of squares and products between covariates, and information about contrasts. By default, Genstat searches all the strata, and takes the information from the lowest of the strata where the term is estimated. If you set the STRATUM option, only strata down to the specified stratum are searched. By setting SUPPRESSHIGHER=yes, you can restrict the search to only that stratum. You cannot save tables of means if you have excluded any stratum from the search. Likewise, tables of residuals and residual sums of squares cannot be saved for any of the excluded strata. If a term is not estimated in any of the strata that are searched, the corresponding data structures are filled with missing values.

The STATUS parameter saves an integer code that describes the type of term, and how it is estimated. If the term is a treatment term, the code also gives information about how its marginal

terms are estimated. (For example, the interaction term `A.B` has the main effects `A` and `B` as margins.)

1	the term is a treatment term; the term itself and all of its margins are orthogonal, and are estimated in the same stratum.
2	the term is a treatment term; the term itself and all of its margins have the same efficiency factor, and are estimated in the same stratum.
3	the term is a treatment term; the term and its margins have different efficiency factors, but are all estimated in the same stratum.
4	the term is a treatment term; the term itself and all of its margins are orthogonal, but are estimated in different strata.
5	the term is a treatment term; the term itself and all of its margins have the same efficiency factor, but are estimated in different strata.
6	the term is a treatment term; the term and its margins have different efficiency factors and are all estimated in different strata.
0	the term is a treatment term; and term itself or one of its margins is aliased.
−1	the term is an orthogonal block term.
−2	the term is a non-orthogonal block term.
*	the term was not in either the block or treatment model but all of its factors occurred somewhere in the analysis (`AKEEP` gives a fault if the term contains factors that did not occur anywhere in the analysis); all other parameters are then ignored for that term.

As explained in the description of the `BLOCKSTRUCTURE` directive, Genstat will set up an extra "factor" denoted `*Units*` if the block formula does not specify the final stratum explicitly. `AKEEP` allows you to refer to this "factor", if necessary, by putting the string `'*Units*'` (or `'*units*'` or `'*UNITS*'`) in the `TERMS` formula. Thus, to save the residual sum of squares in these circumstances, you could put

```
AKEEP '*Units*'; SS=ResidSS
```

Options: `FACTORIAL`, `STRATUM`, `SUPPRESSHIGHER`, `TWOLEVEL`, `RESIDUALS`, `FITTEDVALUES`, `CBRESIDUALS`, `CBCREGRESSION`, `CBCVCOVARIANCE`, `TREATMENTSTRUCTURE`, `BLOCKSTRUCTURE`, `AFACTORIAL`, `WEIGHTS`, `YVARIATE`, `LSDLEVEL`, `AOVTABLE`, `EQFACTORS`, `RMETHOD`, `EXIT`, `SAVE`.

Parameters: `TERMS`, `MEANS`, `SEMEANS`, `SEDMEANS`, `VCMEANS`, `EFFECTS`, `PARTIALEFFECTS`, `REPLICATIONS`, `RESIDUALS`, `DF`, `LSDMEANS`, `DFMEANS`, `SS`, `EFFICIENCY`, `VARIANCE`, `RTERM`, `CEFFICIENCY`, `CREGRESSION`, `CVCOVARIANCE`, `CSSP`, `CONTRASTS`, `XCONTRASTS`, `SECONTRASTS`, `DFCONTRASTS`, `CBMEANS`, `SECBMEANS`, `SEDCBMEANS`, `VCCBMEANS`, `LSDCBMEANS`, `DFCBMEANS`, `CBEFFECTS`, `CBVARIANCE`, `DFCEFFECTS`, `CBCEFFICIENCY`, `STRATUMVARIANCE`, `COMPONENT`, `STATUS`.

See also
Directives: ANOVA, BLOCKSTRUCTURE, COVARIATE, TREATMENTSTRUCTURE.
Procedures:AFMEANS, AUKEEP, A2KEEP, ASPREADSHEET, A2RDA.
Genstat Reference Manual 1 Summary section on: Analysis of variance.

ANOVA

Analyses y-variates by analysis of variance according to the model defined by earlier BLOCKSTRUCTURE, COVARIATE and TREATMENTSTRUCTURE statements.

Options

PRINT = *string tokens*	Output from the analyses of the y-variates, adjusted for any covariates (aovtable, information, covariates, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default aovt, info, cova, mean, miss
UPRINT = *string tokens*	Output from the unadjusted analyses of the y-variates (aovtable, information, effects, residuals, contrasts, means, cbeffects, cbmeans, stratumvariances, %cv, missingvalues); default * i.e. no printing
CPRINT = *string tokens*	Output from the analyses of the covariates, if any (aovtable, information, effects, residuals, contrasts, means, %cv, missingvalues); default * i.e. no printing
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
CONTRASTS = *scalar*	Limit on the order of a contrast of a treatment term; default 4
DEVIATIONS = *scalar*	Limit on the number of factors in a treatment term for the deviations from its fitted contrasts to be retained in the model; default 9
PFACTORIAL = *scalar*	Limit on number of factors in printed tables of means or effects; default 9
PCONTRASTS = *scalar*	Limit on order of printed contrasts; default 9
PDEVIATIONS = *scalar*	Limit on number of factors in a treatment term whose deviations from the fitted contrasts are to be printed; default 9
FPROBABILITY = *string token*	Printing of probabilities for variance ratios (yes, no); default no
PSE = *string token*	Standard errors to be printed with tables of means, PSE=* requests s.e.'s to be omitted (differences, lsd, means); default diff
TWOLEVEL = *string token*	Representation of effects in 2^n experiments (responses, Yates, effects); default resp
DESIGN = *pointer*	Stores details of the design for use in subsequent analyses; default *
WEIGHTS = *variate*	Weights for each unit; default * i.e. all units with weight one
ORTHOGONAL = *string token*	Whether or not design to be assumed orthogonal (notassumed, assumed, compulsory); default nota
SEED = *scalar*	Seed for random numbers to generate dummy variate for determining the design; default 12345
MAXCYCLE = *scalar*	Maximum number of iterations for estimating missing values; default 20
TOLERANCES = *variate*	Allows you to redefine the tolerances for zero used by various parts of the algorithm

NOMESSAGE = *string tokens*	Which warning messages to suppress (nonorthogonal, residual); default *
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
EXIT = *scalar*	Saves an exit code indicating the properties of the design

Parameters

Y = *variates*	Variates to be analysed
RESIDUALS = *variates*	Variate to save residuals for each y variate
FITTEDVALUES = *variates*	Variate to save fitted values
SAVE = *identifiers*	Save details of each analysis for use in subsequent ADISPLAY or AKEEP statements

Description

The ANOVA directive analyses balanced designs. These include most of the commonly occurring experimental designs such as randomized blocks, Latin squares, split plots and other orthogonal designs, as well as designs with balanced confounding, like balanced lattices and balanced incomplete blocks. Many partially balanced designs can also be handled, so a very wide range of designs can be analysed. The necessary condition of *first-order balance* is explained algorithmically by Wilkinson (1970) and Payne & Wilkinson (1976), and mathematically by James & Wilkinson (1971) and Payne & Tobias (1992). However, ANOVA can itself detect whether or not a design can be analysed, so if you are not sure whether or not a particular design is analysable, you can run it through ANOVA and see what happens! (If it is unbalanced, you can use the AUNBALANCED procedure for designs with a single error term, or the REML directive for those with several.)

Before you use ANOVA you must first define the model that is to be fitted in the analysis. Potentially this has three parts. The TREATMENTSTRUCTURE directive specifies the treatment (or *systematic*, or *fixed*) terms for the analysis. The BLOCKSTRUCTURE directive defines the "underlying structure" of the design or, equivalently, the *error* terms for the analysis; in the simple cases where there is only a single error term this can be omitted. The other directive, COVARIATE, lists the covariates if an analysis of covariance is required. At the start of a job all these model-definition directives have null settings. However, once any one of them has been used, the defined setting remains in force for all subsequent analyses in the same job until it is redefined.

The first parameter of ANOVA, Y, lists the variates whose values are to be analysed. Genstat examines them all and forms a list of units for which any of the y-variates or any covariate has a missing value. These units are treated as missing in all the analyses. (This is necessary to avoid having to re-analyse covariates for each y-variate.) However, if your y-variates have different missing units, you may prefer to analyse them with separate ANOVA statements, while saving details of the model and design with the DESIGN option to improve efficiency. Genstat also checks whether any of the y-variates has a restriction. If several variates are restricted, they must all be restricted to the same set of units. Only these units are included in the analysis of each y-variate.

If a y-variate has no values, or if you specify a null entry in the Y list, Genstat produces a *skeleton* analysis-of-variance table, which excludes sums of squares, mean squares and variance ratios; the only other output available is the information summary. You can save a design structure, but no save structure is formed. This is a good way of checking that a design can be analysed, before the experiment is carried out.

The RESIDUALS parameter allows you to specify a variate to save the estimated residuals from each analysis. Genstat will declare this variate for you if you have not done so already. In models where there are several error terms, only the final one is included. Others can be obtained using

the AKEEP directive. The fitted values from the analysis are defined to be the data values minus the estimated residuals. These too can be saved, using the FITTEDVALUES parameter. In models where there are several error terms, only the final error term is subtracted. If this is not what you want, you can save the other error terms using AKEEP and subtract them by CALCULATE.

The last parameter, SAVE, allows you to save the complete details of the analysis in an *ANOVA save structure*. The ADISPLAY directive lets you use a save structure to produce further output. You can also use it in the AKEEP directive to put quantities calculated from the analysis into data structures which you can then use elsewhere in Genstat. Save structures are special compound structures, and Genstat declares them automatically. The save structure for the last y-variate analysed is stored automatically, and forms the default for ADISPLAY and AKEEP if you do not provide one explicitly.

The PRINT option selects which components of output are to be displayed.

aovtable	analysis-of-variance table
information	information summary, giving details of aliasing and non-orthogonality or of any large residuals
covariates	estimates of covariate regression coefficients
effects	tables of estimated treatment parameters
residuals	tables of estimated residuals
contrasts	estimated contrasts of treatment effects
means	tables of predicted means for treatment terms
cbeffects	estimated effects of treatment terms combining information from all the strata in which each term is estimated
cbmeans	predicted means for treatment terms combining information from all the strata in which each term is estimated
stratumvariances	estimated variances of the units in each stratum and stratum variance components
%cv	coefficients of variation and standard errors of individual units
missingvalues	estimates of missing values

The default is intended to give the output that you will require most often from a full analysis: aovtable, information, covariates, means and missingvalues. However, with ANOVA the settings information, covariates and missingvalues will not produce any output unless there is something definite to report.

In analysis of covariance, you can also print output from the analyses of the covariates and from the analysis of the y-variate ignoring the covariates. This is controlled by options CPRINT and UPRINT respectively. These are similar to the PRINT option except that they do have not have the setting covariates, and their defaults are to print nothing.

A table of means is produced by default for each term in the treatment model. By using the PFACTORIAL option you can exclude tables for terms containing more than a specified number of factors; Genstat does not allow tables to have more than nine factors, so the default value of nine gives all the available tables. PFACTORIAL also applies to tables of effects. These are estimates of treatment parameters in the linear model.

The PSE option controls the standard errors printed with the tables of means. The default setting is differences, which gives standard errors of differences of means. The setting means produces standard errors of means, LSD produces least significant differences and by setting PSE=* the standard errors can be suppressed altogether. The significance level to use in the calculation of the least significant differences can be changed from the default of 5% using the LSDLEVEL option.

When a factor has only two levels, Genstat usually prints the difference between the two main

effects instead of the effects themselves. This difference is called a *response*. For interaction terms whose factors all have only two levels, there are two forms of response. The choice between them is controlled by the TWOLEVEL option. If you leave the default, TWOLEVEL=response, Genstat calculates the response for an interaction between two factors as the difference between the two main-effect responses, and so on; this is the form described in most textbooks. By putting TWOLEVEL=Yates, you can obtain the form defined by Yates (1937) in which the responses all have equal standard errors. Alternatively, you can put TWOLEVEL=effects if you prefer not to have responses, but to have the effects themselves, as for factors with more than two levels.

The warnings about any large residuals printed in the information summary can be suppressed by setting the NOMESSAGES option to residuals. The other setting, nonorthogonality, of NOMESSAGES suppresses the warning produced when there is non-orthogonality between treatment terms or covariates.

The treatment terms to be included in the model are controlled by the FACTORIAL option; this sets a limit (by default 3) on the number of factors in a treatment term: terms containing more than that number are deleted.

The CONTRASTS option places a limit on the order of contrast to be fitted. (Contrasts are defined by using the functions POL, REG, COMPARISON, POLND or REGND in the treatment formula.) For a term involving a single factor, the orders of the successive contrasts run from one upwards, with the deviations term (if any) numbered highest. In interactions between contrasts, the order is the sum of the orders of the component parts. The default value for CONTRASTS is 4. Option PCONTRASTS similarly sets a limit on the order of the contrasts that are printed; its default value is 9.

If your design has few or no degrees of freedom for the residual, you may wish to regard the deviations from some of the fitted contrasts as error components, and assign them to the residual of the stratum where they occur. You can do this by the DEVIATIONS option; its value sets a limit on the number of factors in the terms whose deviations are to be retained in the model. For example, by putting DEVIATIONS=1, the deviations from the contrasts fitted to all terms except main effects will be assigned to error. The PDEVIATIONS option similarly controls the printing of deviations: to put PDEVIATIONS=0, for example, would ensure that no deviations are printed. When deviations have been assigned to error, they will not be included in the calculation of tables of means, which will then be labelled "smoothed". However the associated standard errors of the means are not adjusted for the smoothing.

The WEIGHT option allows you to specify a weight for each unit, to define a weighted analysis of variance. You might want to do this if, for example, different parts of the experiment have different variability; each weight would then be proportional to the reciprocal of the expected variance for the corresponding unit. However unless the weights are fairly systematic, for example to give proportional weighted replication, the design is unlikely to be balanced.

Before Genstat does any calculations with the y-variates, it does an initial investigation known as the *dummy analysis* to acquire all the information that it needs for the analysis. You can use the DESIGN option to store this information so that Genstat need not recalculate it for future ANOVA statements. The structure in the option is automatically declared as a pointer if you have not declared it already. It points to several other structures which store information about different aspects of the analysis. The only other details that are required for future analyses are the values of the factors in the block and treatment formulae. If you have not previously declared the design structure, or if it has no values, then the current statement derives and stores the necessary information. If the pointer does already have values, then these are used to do the analysis. In that case, of course, values of the factors in the block and treatment formulae must not have been changed since the design structure was formed. The current settings of options FACTORIAL, CONTRASTS, DEVIATIONS and WEIGHT are then ignored, as is any change in the restrictions on the y-variates. The DESIGN option is particularly useful with designs where there

are many model terms or where there is non-orthogonality, as the dummy analysis may then be time-consuming.

Genstat has a simplified version of the dummy analysis which you can use to save computing time if all the model terms are orthogonal and if, for every term, all the combinations of its factors were applied to the same number of units. A check is incorporated which will detect non-orthogonality except in particularly complicated designs where terms are aliased. If you set option `ORTHOGONAL=assumed`, Genstat does the simple version unless non-orthogonality is detected, whereupon it gives a warning message and then switches to the full version. (Before Release 14, this was requested by setting `ORTHOGONAL=yes`, but the aim now is that options with settings `yes` and `no` do not have any other settings; however, `yes` is retained as a synonym for `assumed`, so that existing programs will still run.) The simplified version is done also if `ORTHOGONAL=compulsory`, but non-orthogonality now causes the analysis to stop altogether, with an error message; this is useful for checking for typing errors in the factor values when you know that the design should otherwise be orthogonal. The dummy analysis involves the analysis of a specially generated variate which contains random numbers from a Cauchy distribution. The starting value for their generation is set by the `SEED` option.

The `TOLERANCES` option controls numerical aspects of analysis. Its setting is a variate with up to four values: the first is used to calculate the tolerance for the analysis of the y-variates (default 10^{-7}), the second is for the tolerance used in the dummy analysis (default 10^{-9}), the third is for the estimation of missing values (default 10^{-5}) and the fourth is for the estimation of stratum variances (default 10^{-5}). The `MAXCYCLE` option sets a limit on the number of iterations for estimating missing values. The `EXIT` option can save an exit code summarizing the properties of the design:

0	design orthogonal;
1	design has general balance (blocks terms mutually orthogonal, treatment terms mutually orthogonal, some treatment terms non-orthogonal to the block terms);
2	blocks terms mutually orthogonal, treatment terms non-orthogonal;
3	block terms non-orthogonal, treatment terms orthogonal;
4	block terms non-orthogonal, treatment terms non-orthogonal;
*	design unbalanced (`ANOVA` failed to analyse it).

Options: `PRINT`, `UPRINT`, `CPRINT`, `FACTORIAL`, `CONTRASTS`, `DEVIATIONS`, `PFACTORIAL`, `PCONTRASTS`, `PDEVIATIONS`, `FPROBABILITY`, `PSE`, `TWOLEVEL`, `DESIGN`, `WEIGHTS`, `ORTHOGONAL`, `SEED`, `MAXCYCLE`, `TOLERANCES`, `NOMESSAGE`, `LSDLEVEL`, `EXIT`.
Parameters: `Y`, `RESIDUALS`, `FITTEDVALUES`, `SAVE`.

Action with `RESTRICT`

You can restrict the set of units used for the analysis by applying a restriction to any of the y-variates. If several are restricted, they must all be restricted to the same set of units. Only these units are included in the analysis of each y-variate.

References

James, A.T. & Wilkinson, G.N. (1971). Factorisation of the residual operator and canonical decomposition of non-orthogonal factors in analysis of variance. *Biometrika*, **58**, 279-294.

Payne, R.W. & Wilkinson, G.N. (1977). A general algorithm for analysis of variance. *Applied Statistics*, **26**, 251-260.

Payne, R.W. & Tobias, R.D. (1992). General balance, combination of information and the analysis of covariance. *Scandinavian Journal of Statistics*, **19**, 3-23.

Wilkinson, G.N. (1970). A general recursive algorithm for analysis of variance. *Biometrika*, **57**, 19-46.

Yates, F. (1937). *The Design and Analysis of Factorial Experiments*. Technical Communication No. 35 of the Commonwealth Bureau of Soils. Commonwealth Agricultural Bureaux, Farnham Royal.

See also

Directives: BLOCKSTRUCTURE, COVARIATE, TREATMENTSTRUCTURE, ADISPLAY, AKEEP, FIT, REML.

Procedures: ABOXCOX, ABLUPS, ACHECK, AFCOVARIATES, AFMEANS, AGRAPH, APLOT, AFIELDRESIDUALS, APERMTEST, APOWER, AMCOMPARISON, AMDUNNETT, AN1ADVICE, APAPADAKIS, APOLYNOMIAL, ARESULTSUMMARY, ASPREADSHEET, ASTATUS, AOVANYHOW, A2RDA, A2WAY, AUNBALANCED, AREPMEASURES, ASCREEN, AYPARALLEL, FALIASTERMS.

Functions: COMPARISON, POL, POLND, REG, REGND.

Genstat Reference Manual 1 Summary sections on: Analysis of variance, Design of experiments, REML analysis of linear mixed models.

ASRULES

Derives association rules from transaction data.

Options

`PRINT` = *string tokens*	Controls printed output (`rules`); default `rule`
`METHOD` = *string tokens*	What to use to calculate the support of a rule (`allitems`, `antecedent`); default `ante`
`MINSUPPORT` = *scalar*	Minimum amount of support for a rule to be included; default 0.1
`MINCONFIDENCE` = *scalar*	Minimum amount of confidence for a rule to be included; default 0.8
`MAXITEMS` = *scalar*	Maximum number of items that a rule may contain; default 10
`MAXRULES` = *scalar*	Maximum number of rules to generate; default 100

Parameters

`ITEMS` = *factors*	Items in the transactions
`TRANSACTIONS` = *factors*	Specifies the transaction to which each each item belongs
`NRULES` = *scalars*	Saves the number of rules that have been derived
`RULES` = *pointers*	Pointer to factors, each of which saves the antecedent items and then the consequent item in one of the rules
`SUPPORT` = *variates*	Saves the support values for the rules
`CONFIDENCE` = *variates*	Saves the confidence values for the rules

Description

`ASRULES` examines a set of "transaction data" to derive rules of the form: "if a transaction contains items a_1 ... a_m, then it is likely also to contain item c". The items a_1 ... a_m are known as the *antecedent* set, and the item c is known as the *consequent* item.

The data are specified in a pair of factors, using the `ITEMS` and `TRANSACTIONS` parameters. `ITEMS` specifies the items involved in (all) the transactions, and `TRANSACTIONS` specifies the transaction to which each item belongs. The data must be provided in sorted order, one transaction at a time and the items within each transaction in ascending order. For example

```
Items        2 3 5 1 6 7 8 4 7 9 10 1 3 4 6 8 ...
Transactions 1 1 1 2 2 2 2 3 3 3  3 4 4 4 4 4 ...
```

You can do this with the `SORT` directive. For example, if the transactions factor is `Trans` and the items factor is `Items`, the command would be

```
SORT [INDEX=Trans,Items] Trans,Items
```

`ASRULES` finds sets of items that occur frequently together within the transactions, and then examines these to derive the rules.

The *support* of a set of items is the proportion of the transactions that contains them. To avoid presenting rules that have little justification, the `MINSUPPORT` option defines a minimum value for the support of a rule for it to be included (default 0.1). The `METHOD` option controls whether the support is defined to be the support for all the items in the rule, or only of its antecedent items (the default).

The *confidence* of a rule, is the proportion of those transactions that contain the antecedent set of items that also contains the consequent set. The `MINCONFIDENCE` option a minimum value for the confidence of a rule for it to be included (default 0.8).

The `MAXITEMS` option sets a maximum limit on the number of items that a rule may contain (default 10), and the `MAXRULES` option specifies the maximum number of rules that may be generated (default 100).

By default the rules are printed, with their support and confidence values. However, this can be suppressed by setting option PRINT=*.

The number of rules that have been derived can be saved, in a scalar, using the NRULES parameter. The rules themselves can be saved using the RULES parameter, in a pointer to a set of factors. Each factor contains the the antecedent items and then the conseqent item for a particular rule. The SUPPORT parameter can save a variate with a unit for each rule, containing its support. The CONFIDENCE parameter similarly saves the confidence of the rules.

Options: PRINT, METHOD, MINSUPPORT, MINCONFIDENCE, MAXITEMS, MAXRULES.
Parameters: ITEMS, TRANSACTIONS, NRULES, RULES, SUPPORT, CONFIDENCE.

Method
ASRULES uses the function nagdmc_assoc from the Numerical Algorithms Group's library of Data Mining Components (DMCs).

Action with RESTRICT
ITEMS and TRANSACTIONS may be restricted to derive the rules from only a subset of the data.

See also
Directives: NNFIT, RBFIT.
Procedure: KNEARESTNEIGHBOURS.
Genstat Reference Manual 1 Summary section on: Data mining.

ASSIGN

Sets elements of pointers and dummies.

Options

NSUBSTITUTE = *scalar*	Number of times *n* to substitute a dummy setting of the POINTER parameter in order to determine which dummy should be assigned the setting of the STRUCTURE parameter (if *n* is negative, the assigned dummy is the −*n*th from the bottom of the chain of dummies, like the NTIMES option of EXIT); default 0 i.e. no substitution
METHOD = *string token*	Whether to replace or preserve the existing value in each dummy or pointer element (replace, preserve); default repl (note, pointer elements are never unset so METHOD=preserve with a pointer simply causes the assignment to be ignored)
RENAME = *string token*	Whether to reset the default name for the structure if it has only a suffixed identifier (yes, no); default no
SCOPE = *string token*	This allows dummies or pointer elements within a procedure to be set to point to structures in the program that called the procedure (SCOPE=external) or in the main program itself (SCOPE=global) rather than to structures within the procedure (local, external, global); default loca
NSTRUCTURESUBSTITUTE = *scalar*	Number of times *n* to substitute a dummy setting of the STRUCTURE parameter in order to determine which structure to assign to the setting of the POINTER parameter (if *n* is negative, the assigned structure is the −*n*th from the bottom of the chain of dummies, like the NTIMES option of EXIT); default 0 i.e. no substitution

Parameters

STRUCTURE = *identifiers*	Values for the dummies or pointer elements
POINTER = *dummies* or *pointers*	Structure that is to point to each of those in the STRUCTURE list
ELEMENT = *scalars* or *texts*	Unit or unit label indicating which pointer element is to be set; if omitted, the first element is assumed

Description

ASSIGN allows you to set individual elements of pointers, or to assign a value to a dummy. The parameter POINTER lists the pointers or dummies whose values you want to set; the values that you want to give them are listed by the STRUCTURE parameter. You pick out the individual elements of pointers by the ELEMENT parameter; a scalar identifies the element by its suffix number, while a text identifies it by its label. This example sets the dummy Yvar to point to the variate Height, and elements 1 and 2 of the pointer Xvars to Protein and Vitamin, respectively.

```
VARIATE Height,Protein,Vitamin
POINTER [NVALUES=2] Xvars
DUMMY Yvar
ASSIGN Height,Protein,Vitamin;POINTER=Yvar,2(Xvars); \
  ELEMENT=1,1,2
```

Element 1 is assumed unless you specify otherwise; so to set just `Yvar` we need only put

```
ASSIGN Height; POINTER=Yvar
```

Options `NSUBSTITUTE` and `METHOD` are likely to be most useful when setting dummies within a procedure. By setting `METHOD=preserve`, any dummies that are already set will have their existing settings preserved. Hence this provides a very convenient and effective way of making default assignments while leaving any explicit assignments unchanged. Suppose, for example, that a procedure has dummy arguments `FITTEDVALUES`, `RESIDUALS` and `RSS` available to save various aspects of the analysis, and that we wish to use these as working variables while calculating this information within the procedure. By specifying

```
ASSIGN [METHOD=preserve] LocalF,LocalR,LocalRSS; \
  FITTEDVALUES,RESIDUALS,RSS
```

any of the dummies that is not set when the procedure is called will be assigned to the corresponding local structure, either `LocalF`, `LocalR` or `LocalRSS`. Note, however, that elements of pointers cannot be unset; they will always point to some identifier, even if it is unnamed. Thus, `ASSIGN` has no effect on elements of pointers when `METHOD=preserve`.

The `NSUBSTITUTE` option is useful when you have dummies pointing to other dummies, in a chain. This can often happen when one procedure calls another, passing one of its own arguments as the argument to the procedure that it calls. A positive setting substitutes the dummies in the `POINTER` list the defined number of times in order to determine which dummy in a chain is to be assigned a value. Alternatively, you can set `NSUBSTITUTE` to a negative integer to specify the dummy to assign by counting up from the bottom of the chain of dummies, instead of down from the top.

Similarly, the `NSTRUCTURESUBSTITUTE` option is useful when you have a dummy as the setting of the `STRUCTURE` parameter. By default, it is the dummy itself that is assigned to the corresponding dummy or pointer in the `POINTER` list. However, you can set `NSTRUCTURESUBSTITUTE`, in the same way as `NSUBSTITUTE`, to substitute the dummy before making the assignment.

The `RENAME` option enables you to control what identifier is used for data structures in the rare occasions when your program contains structures that can be referred to by more than one suffixed identifier and which do not have identifiers in their own right.

Finally, the `SCOPE` option enables you to assign a dummy within a procedure to a structure in the program that called the procedure. The dummy will thus operate as though it was a dummy option or parameter, except that the decision about the structure that it references in the outer program has been made within the procedure instead of outside it. This facility allows you to define new data structures in the outer program; however, care needs to be taken to ensure that there is no conflict with any existing structures.

Options: `NSUBSTITUTE`, `METHOD`, `RENAME`, `SCOPE`, `NSTRUCTURESUBSTITUTE`.
Parameters: `STRUCTURE`, `POINTER`, `ELEMENT`.

See also

Directives: `DUMMY`, `POINTER`, `PROCEDURE`.

Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Program control.

AXES

Defines the x- and y-axis in each window for high-resolution graphics.

Options

EQUAL = *string tokens*	Whether/how to make axes equal (no, scale, lower, upper); default no
RESET = *string token*	Whether to reset the axes definitions to the default values (yes, no); default no

Parameters

WINDOW = *scalars*	Numbers of the windows
YTITLE = *texts*	Title for the y-axis in each window
XTITLE = *texts*	Title for the x-axis in each window
YLOWER = *scalars*	Lower bound for y-axis
YUPPER = *scalars*	Upper bound for y-axis
XLOWER = *scalars*	Lower bound for x-axis
XUPPER = *scalars*	Upper bound for x-axis
YMARKS = *scalars* or *variates*	Distance between each tick mark on y-axis (scalar) or positions of the marks (variate)
XMARKS = *scalars* or *variates*	Distance between each tick mark on x-axis (scalar) or positions of the marks (variate)
YMPOSITION = *string tokens*	Position of the tick marks across the y-axis (left, right, centre)
XMPOSITION = *string tokens*	Position of the tick marks across the x-axis (above, below, centre)
YLABELS = *texts*	Labels at each mark on y-axis
XLABELS = *texts*	Labels at each mark on x-axis
YLPOSITION = *string tokens*	Position of the labels for the y-axis (left, right)
XLPOSITION = *string tokens*	Position of the labels for the x-axis (above, below)
YORIGIN = *scalars*	Position on y-axis at which x-axis is drawn
XORIGIN = *scalars*	Position on x-axis at which y-axis is drawn
STYLE = *string tokens*	Style of axes (none, x, y, xy, box, grid)
PENTITLE = *scalars*	Pen to use for the title
PENAXES = *scalars*	Pen to use for the axes and their labelling
PENGRID = *scalars*	Pen to use for the grid
SAVE = *pointers*	Saves details of the current settings for the axes concerned

Description

There is a definition for the x- and y-axis associated with each Genstat graphics window. This specifies how the axes are to be drawn when graphical output is produced in that window. The default definition for each set of axes requires some of the features to be determined from the data, as described below. Others have fixed defaults that are independent of the data. The AXES directive can be used to override the default action and specify explicitly how particular parts of the axis are drawn. All parameters of AXES are relevant when using DGRAPH, but for other directives only some of the parameters are used.

The graphical attributes controlled by AXES can be set more conveniently using the directives XAXIS, YAXIS and ZAXIS, and extensions to the FRAME directive, that were introduced in Release 4.2. AXES has been retained to enable existing programs to run, but it is recommended that for new programs FRAME, XAXIS, YAXIS and ZAXIS be used instead.

The WINDOW parameter specifies the window whose axes definitions are to be altered. WINDOW

can be set to a list of window numbers, in which case the other parameter lists are cycled in the usual way. By default, only those aspects specified by subsequent parameter lists are modified; any parameters that are not set will retain their current settings. Alternatively, you can specify option `RESET=yes` to reset the values of any parameters that are not set for each window, back to the default values that are set up by Genstat at the start of a job.

The `YLOWER` and `YUPPER` parameters specify the lower and upper bounds for the y-axis. By default, Genstat derives suitable axis bounds from the data, as described for the appropriate directive. You can set the lower bound to a value greater than the upper bound, to obtain an inverted data scale, but the bounds must not be equal. The `XLOWER` and `XUPPER` parameters set bounds for the x-axis in a similar way. The values specified with these parameters are on the scale of the data values that are plotted, and are independent of the normalized device coordinates used to define the window size in `FRAME`. The `EQUAL` option can be used to ensure that equal upper or lower bounds are used for the y- and x-axes. For example, if `EQUAL=lower`, lower bounds for both axes will be set to the lower of the values determined automatically from the data. The bounds obtained when using the `EQUAL` option may be constrained by settings of other parameters: for example, if `YUPPER` is set and `EQUAL=upper`, the upper bounds of both axes are set to the value specified by `YUPPER`; but if `XUPPER` is also set, `EQUAL` will be ignored. You can set `EQUAL=lower,upper` to constrain both upper and lower bounds, and `EQUAL=scale` can be used to ensure physical distance is equal on both axes, for example the y-axis could range from 0 to 100 and the x-axis from 100 to 200.

The `YORIGIN` parameter determines the value on the y-axis through which the x-axis is drawn. If its value is outside the y-axis bounds, the upper or lower bound is adjusted so that the axis will extend up to the specified origin. This applies whether you have set the bounds explicitly or have left Genstat to calculate them from the data. The `XORIGIN` parameter sets the origin for the x-axis in a similar way. By default, the lower bounds of each axis are used, so that the axes are drawn on the bottom and left-hand sides of the plot.

Titles can be added to the axes using the `YTITLE` and `XTITLE` parameters. In each case, the title is limited to a single line of characters.

Each axis is marked with a scale, determined automatically so that tick marks are evenly spaced and positioned to give "round" numbers for the scale values. For each axis, you can specify either the increment between tick marks or their actual positions. You can also specify labels to use for scale markings instead of their numerical values.

To specify the increment on the y-axis, the `YMARKS` parameter should be set to a scalar. For example, `YMARKS=1.5` with bounds 10 and 2 causes tick marks to appear at 2, 3.5, 5, 6.5, 8 and 9.5. The interval must be a positive number, irrespective of the values of the bounds. Alternatively, you can set `YMARKS` to a variate (with more than one value) to specify the actual positions of the tick marks on the y-axis. Any values that lie outside the axis bounds are ignored. The scale values printed next to the tick marks use a format that is determined automatically from the values, but if you have set `YMARKS` to a variate it will use the number of decimals specified in the variate declaration. When you have set `YMARKS`, you can also use the `YLABELS` parameter to specify a set of labels to mark the axis scale. For example,

```
TEXT [VALUES=Mon,Tues,Wed,Thur,Fri,Sat,Sun] Day
VARIATE [VALUES=1...31] Month
AXES 1; YMARKS=Month; YLABELS=Day
```

The strings within the text are cycled if necessary; hence, the number of strings can be less than the number of tick marks.

The tick marks can be drawn to the left or to the right of the axis, or can be centred (that is, across the axis). By default, the tick marks are drawn towards the "outside" of the plot; that is, to the left if the y-axis is to the left of the centre of the plot, or to the right if the y-axis is drawn to the right of centre. The aim is to position the tick marks away from the main part of the plot, so that they interfere with the plotted points as little as possible. You can control the positioning

of the tick marks by setting the YMPOSITION parameter to either left, right or centre. A similar rule governs the default positioning of the scale markings or labels, but you can again control this by setting the YLPOSITION parameter to either left or right. Setting YMARKS=* will return to the default positioning of the tick marks; YLABELS=* will switch off any labels previously specified; and YMPOSITION=* and YLPOSITION=* will switch off tick marks or labels altogether.

Annotation of the x-axis can be controlled in a similar way using the XMARKS, XLABELS, XMPOSITION and XLPOSITION parameters, except that the settings left and right are replaced by above and below.

The STYLE parameter controls the type of axes that are drawn. By default STYLE=xy, so both y- and x-axes are plotted. Alternative settings allow the axes to be completed by drawing a box around the graph, with an overlaid grid if required. The settings STYLE=x and STYLE=y can be used if only one axis is required. Finally, STYLE=none inhibits the plotting of axes completely, although some other parameters, such as YLOWER, may still have an effect on the plotted data.

There are three parameters that control the pens to be used when drawing the axes. These are PENTITLE, PENAXES and PENGRID, specifying the pen for the title, the axes and annotation, and the grid, respectively. The initial default is to use pens -1, -2 and -4 in every window. These pens are given negative numbers to allow them to be distinguished from the pens used for the contents of the plot. They are initially set up to use colour 1, line style 1, thickness 1, size 1 and font 1. You can thus control which pens are used for drawing the axes in each window, and the attributes of those pens. For example, if no AXES statement has yet been given,

```
PEN -4; LINESTYLE=4; COLOUR=2
```

will request that the grids in every window should be drawn in line style 4 and colour 2; while

```
PEN 29; LINESTYLE=3; COLOUR=4
AXES 1; PENAXES=29
```

will change the appearance of just the axes in window 1, as pen 29 is not used for the other windows. Control of the grid pen is particularly useful as a combination of colour and line style can be chosen to ensure that the grid does not obscure the plotted points. You should of course be careful of side-effects when changing the pen numbers. For example, pen 29 may also have been modified for use in a DGRAPH statement and other attributes may have been set that are not wanted when drawing the axes.

Axis annotation is plotted in the margins specified by the FRAME directive. You may wish to reduce the size of these margins if you have defined axes that use less space, for example by keeping within the area of the graph itself, or by omitting titles or labels. Space can thus be regained and used for plotting data. However, if the margins are too small the axis annotation may be "clipped" at the boundaries of the margins; if this happens, you can use FRAME to increase the margin size. The margins are used by DGRAPH, DHISTOGRAM and DCONTOUR, but they are ignored by other directives.

The current settings of the axes for a particular window can be saved in a pointer supplied by the SAVE parameter. The elements of the pointer are labelled to identify the components. This facility is of most use within procedures, where it may be necessary to check or modify particular AXES settings before constructing complicated graphs. Also, the DKEEP directive allows you to extract the actual bounds used when plotting; these will be the bounds determined from the data if none have been defined explicitly by AXES.

Options: EQUAL, RESET.

Parameters: WINDOW, YTITLE, XTITLE, YLOWER, YUPPER, XLOWER, XUPPER, YMARKS, XMARKS, YMPOSITION, XMPOSITION, YLABELS, XLABELS, YLPOSITION, XLPOSITION, YORIGIN, XORIGIN, STYLE, PENTITLE, PENAXES, PENGRID, SAVE.

See also
Directives: AXIS, XAXIS, YAXIS, ZAXIS.
Genstat Reference Manual 1 Summary section on: Graphics.

AXIS

Defines an oblique axis for high-resolution graphics.

Option

RESET = *string token*	Whether to reset the axis definition to the default values (yes, no); default no

Parameters

IDENTIFIER = *identifiers*	Name to be used inside Genstat to identify each axis
TITLE = *texts*	Title for each axis
TPOSITION = *string tokens*	Position of title (middle, end)
TDIRECTION = *string tokens*	Direction of title (parallel, perpendicular)
LOWER = *scalars*	Lower bound for each axis
UPPER = *scalars*	Upper bound for each axis
MARKS = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along each axis (variate)
MPOSITION = *string tokens*	Positioning of the tick marks on each axis (inside, outside, across)
LABELS = *texts* or *variates*	Labels at each major tick mark
LPOSITION = *string tokens*	Position of the axis labels (inside, outside)
LDIRECTION = *string tokens*	Direction of the axis labels (parallel, perpendicular)
LROTATION = *scalars* or *variates*	Rotation of the axis labels
NSUBTICKS = *scalars*	Number of subticks per interval (ignored if MARKS is a variate)
XZERO = *scalars*	Position of the axis origin in the x-dimension
YZERO = *scalars*	Position of the axis origin in the y-dimension
ZZERO = *scalars*	Position of the axis origin in the z-dimension
XSTEP = *scalars*	Step in the x-direction corresponding to a step of length one along the axis
YSTEP = *scalars*	Step in the y-direction corresponding to a step of length one along the axis
ZSTEP = *scalars*	Step in the z-direction corresponding to a step of length one along the axis
PENTITLE = *scalars*	Pen to use to write the axis title
PENAXIS = *scalars*	Pen to use to draw the axis
PENLABELS = *scalars*	Pen to use to write the axis labels
ARROWHEAD = *string tokens*	Whether the axis should have an arrowhead (include, omit)
ACTION = *string tokens*	Whether to display or hide the axis (display, hide)
TRANSFORM = *string tokens*	Transformed scale for the axis marks and labels (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
DREPRESENTATION = *scalars* or *variates*	Format to use for dates and times printed at the marks
VREPRESENTATION = *string tokens*	Format to use for numbers printed at the marks (decimal, engineering, scientific); default deci
ZEROOFFSET = *scalars*	Point on the axis corresponding to XZERO, YZERO and

	ZZERO
SAVE = *pointers*	Saves details of the current settings for the axis concerned

Description

The AXIS directive allows you to define an oblique axis for high-resolution graphics. You use the IDENTIFIER parameter to supply an identifier to store the axis definition. You can then use this as a setting of the AXES parameter of the FRAME directive to display the axis in a particular graphics window. The other parameters define particular attributes of the axis. Any that are not set in a particular AXIS statement retain their existing settings. These may be the initial default settings, which are the same as those of other (x-, y- or z-) axes, or they may have been defined by an earlier AXIS statement with the same axis identifier. Alternatively, you can set option RESET=yes to reset the values back to the initial default values.

The position of the axis in the x-, y- and z-dimensions of the window is specified by the parameters XZERO, YZERO and ZZERO, respectively. The initial default is for these to define the position of the origin of the axis i.e. the zero point. However, you can use the ZEROOFFSET parameter to specify that the position is at another point along the axis. This may be more accurate, if the axis origin is a long way outside the graphics frame. The XSTEP, YSTEP and ZSTEP parameters define the size of the steps in the x-, y- and z-directions that corresponds to a step of length one along the axis. These seven parameters thus define the location and direction of the axis.

You can specify a title for the axis using the TITLE parameter. This is limited to a single line of characters. The TPOSITION parameter controls whether the title is placed in the middle or at the end of the axis, and the TDIRECTION parameter controls whether it is written parallel or perpendicular to the axis.

The LOWER and UPPER parameters specify the lower and upper bounds for the axis. By default, Genstat sets the axis bounds so that the axis goes from one side of the window to the other.

The axis is marked with a scale, determined automatically so that tick marks are evenly spaced and positioned to give "round" numbers for the scale values. You can set the MARKS parameter to a scalar to define the increment between tick marks. For example, setting MARKS=1.5 with bounds 10 and 2, causes tick marks to appear at 2, 3.5, 5, 6.5, 8 and 9.5. The interval must be a positive number, irrespective of the values of the bounds. Alternatively, you can set MARKS to a variate (with more than one value) to specify the actual positions of the tick marks on the axis. Any values that lie outside the axis bounds are ignored. The scale values printed next to the tick marks use a format that is determined automatically from the values, but if you set MARKS to a variate it will use the number of decimals specified in the variate declaration. If MARKS is unset or set to a scalar, you can use the NSUBTICKS parameter to specify a number of "subticks" to be drawn between each of the (major) tick marks.

When you set MARKS, you can also use the LABELS parameter to specify a set of labels to print at the (major) axis marks, instead of the numbers. For example,

```
TEXT [VALUES=Mon,Tues,Wed,Thur,Fri,Sat,Sun] Day
VARIATE [VALUES=1...31] Month
AXIS Timeax; MARKS=Month; LABELS=Day
```

The strings within the text are cycled if necessary, so the number of strings can be less than the number of tick marks. The DECIMALS parameter can set the number of decimal places to use if you are printing numbers at the marks. If the numbers represent dates or times, you should specify their formats using the DREPRESENTATION parameter (see the PRINT directive for details). By default, numbers are printed in decimal form. If you would prefer scientific format you can set parameter VREPRESENTATION=scientific; numbers are then printed as a decimal number with absolute value less than 10, followed by an exponent (e.g. 3.4567 E4 for 34567). Alternatively, you can set VREPRESENTATION=engineering to use engineering format; the

decimal number then has an absolute value less than 10000, so the exponent is a multiple of 3 (e.g. 34.567 E3 for 34567). With scientific or engineering formats, the DECIMALS parameter sets the number of significant figures to use rather than the number of decimal places.

The MPOSITION parameter controls the positioning of the tick marks, which can be drawn on the inside or the outside of the axis, or can be drawn across the axis. With the outside setting, the tick marks are drawn towards the outside of the plot; that is below the axis if the axis is in the lower half of the plot, or above the axis if it is in the top half of the plot. The aim is then to position the tick marks away from the main part of the plot, so that they interfere with the plotted points as little as possible. With the inside setting, the marks are drawn on the opposite side (that is, to the inside of the plot), while the across setting draws them across the axis. Similarly, the positioning of the scale markings or labels is controlled by the LPOSITION parameter, with settings inside or outside. The LDIRECTION parameter controls whether the scale markings or labels are written parallel or perpendicular to the axis. Alternatively, you can use the LROTATION parameter to specify the direction of the labels more precisely, as a rotation in degrees from the horizontal (i.e. parallel) direction. If LROTATION is specified, any setting of LDIRECTION is ignored.

Setting MARKS=* will return to the default positioning of the tick marks. Setting LABELS=* will switch off any labels previously specified. Setting MPOSITION=* will switch off any tick marks, and setting LPOSITION=* or LDIRECTION=* will switch off any labels.

The TRANSFORM parameter allows you to transform the marks and labels on the axis. The location and direction of the axis are defined according to the original scale, by the XZERO, YZERO, ZZERO, ZEROOFFSET, XSTEP, YSTEP and ZSTEP parameters, as usual. The coordinates along the axis are then transformed, and labelled according to the transformed scale. So, for example, with TRANSFORM=log10, the original coordinates 1, 10 and 100 would be labelled 0, 1 and 2. The settings are the same as the names of the equivalent Genstat functions, with the addition of exp10 for the antilog transformation (i.e. 10^x), and square for x^2.

There are three parameters to control the pens to be used to draw the axis. These are PENTITLE, PENAXIS and PENLABEL, specifying the pen for the title, the axis and the labelling, respectively. The initial default is to use pens −1, −2 and −3 as for other axes.

The ARROWHEAD parameter controls whether the axis is drawn with an arrowhead at the end. The ACTION parameter controls whether or not the axis is displayed or hidden initially when the window is used for a plot (you can then choose to display the axis from within the graphics viewer).

The current settings defined for the axis can be saved in a pointer supplied by the SAVE parameter. The elements of the pointer are labelled to identify the components.

Option: RESET.

Parameters: IDENTIFIER, TITLE, TPOSITION, TDIRECTION, LOWER, UPPER, MARKS, MPOSITION, LABELS, LPOSITION, LDIRECTION, LROTATION, NSUBTICKS,, XSTEP, YSTEP, ZSTEP, PENTITLE, PENAXIS, PENLABELS, ARROWHEAD, ACTION, TRANSFORM, DECIMALS, DREPRESENTATION, VREPRESENTATION, ZEROOFFSET SAVE.

See also

Directives: XAXIS, YAXIS, ZAXIS, FRAME.

Genstat Reference Manual 1 Summary section on: Graphics.

BARCHART

Plots bar charts in high-resolution graphics.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the bar charts; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
BARWIDTH = *scalar*, *variate* or *table*	Width(s) of the bars; default * sets equal widths to fill the x-axis
BARCOVERING = *scalar*	What proportion of the space allocated along the x-axis each bar should occupy; default * gives proportion 1 for a DATA variate, and 0.8 for a factor or table (thus giving a gap between each bar)
LABELS = *text*	Labels for the bars or groups of bars; default *
APPEND = *string token*	Whether or not the bars of the bar charts are appended together (yes, no); default no
ORIENTATION = *string token*	Direction of the plot (horizontal, vertical); default vert
YSCALING = *string token*	What scale to use to label the y-axis (absolute, proportion, percentage); default abso
OUTLINE = *string token*	Where to draw outlines (bars, perimeter); default bars
PENOUTLINE = *scalar*	Pen to use for the outlines; default -9
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

DATA = *tables* or *variates*	Heights of the bars in each bar chart
ERRORBARS = *scalars*, *tables* or *variates*	Heights of error bars plotted above the bars of each bar chart; default 0 i.e. none
LOWERERRORBARS = *scalars*, *tables* or *variates*	Heights of error bars plotted below the bars of each bar chart; if any of these is omitted, the corresponding setting of ERRORBARS is used as the default so that the error bars will have equal heights above and below the bars of the bar chart
GROUPS = *factors*	Which factor of a 2-way table to use as the groups factor; default uses the second classifying factor
PEN = *scalars*, *tables* or *variates*	Pen number(s) for each bar chart; default * uses pens 2, 3, and so on for the successive structures specified by DATA
PENERRORBARS = *scalars*, *tables* or *variates*	Pen number(s) for the error bars; default -11
DESCRIPTION = *texts*	Annotation for key

Description

BARCHART plots high-resolution bar charts. You can plot a single bar chart by setting the DATA parameter to a one-way table or a variate defining the heights of the bars. To plot several bar charts on the same graph, you can set DATA to a list of one-way tables or variates. These must all contain the same number of values, and any tables must be classified by the same factor.

Alternatively, you can set DATA to a two-way table. The GROUPS parameter then specifies which of the two classifying factors is to be treated as the "groups" factor (by default this is the second factor). BARCHART now plots a bar chart for every level of the GROUPS factors, with bars defined by the other classifying factor.

Labels can be supplied for the bars, using the LABELS option. If this is not set, the labels will be the labels or levels of the factor classifying the DATA tables, or the integers 1 upwards for a DATA variate.

By default, if there are several bar charts, they are plotted with their bars alongside each other. So BARCHART first plots the first bar of every bar chart, then the second bar, and so on. Alternatively, you can set option APPEND=yes to stack the bars. The bottom portion of each bar then corresponds to the first bar chart, and the top to the last bar chart.

You can include error bars in a single bar chart or when several bar charts are plotted alongside each other, by specifying their heights with the ERRORBARS and LOWERERRORBARS parameters. The error bars take the form of a horizontal line joined by a vertical line of the specified height, above and below each bar. The ERRORBARS parameter specifies the heights of the error bars above the bars of the bar chart, and the LOWERERRORBARS parameter specifies the heights of the error bars below the bars. If LOWERERRORBARS is not specified, the error bars are assumed to have the same heights below and above the bars. You can set ERRORBARS and LOWERERRORBARS to a scalar if the heights are the same for every bar of a bar chart, or to a table or variate if different bars have error bars with different heights.

The ORIENTATION option controls whether the bars of the bar chart are plotted vertically (the default) or horizontally. When ORIENTATION=horizontal, the horizontal axis is taken to be the y-axis, so the same XAXIS and YAXIS settings can be used however the bar chart is oriented.

By default, Genstat uses pen 2 for the first bar chart, pen 3 for the second bar chart, and so on, so that a different colour is used for each one. Alternatively, you can define your own colours or shading, using the PEN parameter. If you set PEN to a scalar, a single pen is used for all the bars. Alternatively, you can specify a variate or a table to define a different pen for each bar. The relevant aspects of the pens should be set in advance, if required, using the COLOUR parameters of the PEN directive. Generally, however, the default attributes of the pens will be satisfactory. Similarly, the PENERRORBARS parameter specifies the pen or pens to use for the error bars (default −11).

The bars in a bar chart usually have equal widths, defined to fill the available space along the x-axis. However, you can set your own widths by setting option BARWIDTH to either a scalar or a variate or table with as many values as the number of bars. The BARCOVERING option indicates what proportion of the space allocated along the x-axis each bar should occupy; the default is 0.8 (giving a gap between each bar).

The OUTLINE option controls whether lines are drawn around the bars or around the perimeter of the bar chart. These are drawn using the pen specified by the PENOUTLINE option (default −9). You can suppress all the outlines by setting OUTLINE=*.

The WINDOW option defines the window where the bar chart is plotted, and the KEYWINDOW option similarly specifies where the key should appear. You can set either of these to zero if you want to suppress the corresponding output. Titles can be added to the bar chart and key using the TITLE and KEYDESCRIPTION options respectively.

The SCREEN option controls whether the graphical display is cleared before the bar chart is plotted and the ENDACTION option controls whether Genstat pauses at the end of the plot.

The axes of the plot are formed automatically from the data. By default, the upper bound of

the y-axis is set to be five percent greater than the height of the longest bar. If any of the bars has a negative height the lower bound is adjusted in a similar way, otherwise it is set to zero. You can control the form of the axes by using the XAXIS and YAXIS directives to set the required attributes (such as titles) before the BARCHART directive is used. The YSCALING option controls the scale used to label the y-axis, with settings absolute, proportion or percentage; the default is absolute.

The key consists of the title, if set by KEYDESCRIPTION, followed information about each bar chart. You can specify a description for each bar chart using the DESCRIPTION parameter. If the DATA parameter was set to a list of one-way tables or variates, the default description takes the identifier of the table or variate. If DATA was set to a two-way table, the default descriptions are formed from the labels or levels of the GROUPS factor.

Options: TITLE, WINDOW, KEYWINDOW, BARWIDTH, BARCOVERING, LABELS, APPEND, ORIENTATION, SCALING, OUTLINE, PENOUTLINE, SCREEN, KEYDESCRIPTION, ENDACTION.
Parameters: DATA, ERRORBARS, LOWERERRORBARS, GROUPS, PEN, PENERRORBARS, DESCRIPTION.

See also
Directives: DHISTOGRAM, D3HISTOGRAM, DPIE, LPHISTOGRAM, FRAME, XAXIS, YAXIS, PEN.
Procedures: TRELLIS, DBARCHART, DMASS, DOTHISTOGRAM, DOTPLOT, DCIRCULAR, WINDROSE.
Genstat Reference Manual 1 Summary section on: Graphics.

BASSESS

Assesses potential splits for regression and classification trees.

Options

`Y` = *variate* or *factor*	Response variate for a regression tree, or factor specifying the groupings for a classification tree
`SELECTED` = *dummy*	Returns the identifier of `X` variate or factor used in the best split
`TESTSPLIT` = *expression structure*	Logical expression representing the best split
`MAXSPLITPOINT` = *scalar* or *variate*	When `SELECTED` is a variate or a factor with ordered levels this returns a scalar containing the boundary between the two splits, when the `SELECTED` is a factor with unordered levels it returns a variate containing the levels allocated to the first split
`MAXCRITERION` = *scalar*	Maximum value obtained for the selection criterion
`NOSELECTION` = *scalar*	Returns the value 1 if no split has been selected, otherwise 0
`FMETHOD` = *string token*	Selection method to use when `Y` is a factor (`Gini`, `MPI`); default `Gini`
`ANTIENDCUTFACTOR` = *string token*	Anti-end-cut factor to use when `Y` is a factor (`classnumber`, `reciprocalentropy`); default `*` i.e. none
`WEIGHTS` = *variate*	Weights; default `*` i.e. all weights 1
`TOLERANCE` = *scalar*	Tolerance multiplier used e.g. to check for equality of x-values; default `*` i.e. set automatically for the implementation concerned

Parameters

`X` = *variates* or *factors*	Variables available to make the split
`ORDERED` = *string tokens*	Whether factor levels are ordered (`yes`, `no`); default `no`
`SPLITPOINT` = *scalars* or *variates*	Saves details of the best split found for each `X` variable; when `X` is a variate or a factor with ordered levels this returns a scalar containing the boundary between the two splits, when the `X` is a factor with unordered levels it returns a variate containing the levels allocated to the first split
`CRITERIONVALUE` = *scalars*	Saves the value of the selection criterion for the best split found for each `X` variable

Description

`BASSESS` selects splits for use when constructing classification or regression trees. The `Y` option specifies the factor defining the groupings for a classification tree, or the response variate for a regression tree. The x-variables that are available to make the split are supplied by the `X` parameter. They can be variates, or factors with either ordered or unordered levels as indicated by the `ORDERED` parameter. For example, a factor called `Dose` with levels for example 1, 1.5, 2 and 2.5 would usually be treated as having ordered levels, whereas levels labelled `'Morphine'`, `'Amidone'`, `'Phenadoxone'` and `'Pethidine'` of a factor called `Drug` would be regarded as unordered.

In a regression tree, the accuracy of each node is the squared distance of the values of the

response variate from their mean for the observations at the node, divided by the total number of observations. The potential splits are assessed by their effect on the accuracy, that is the difference between the initial accuracy and the sum of the accuracies of the two successor nodes resulting from the split.

For a classification tree, the FMETHOD option allows one of two selection criteria to be requested, either Gini information or the MPI (*mean posterior improvement*) criterion of Taylor & Silverman (1993). The default is to use Gini information. The ANTIENDCUTFACTOR option allows you to request use of adaptive anti-end-cut factors as devised by Taylor & Silverman (1993, Section 5). Further details are given in the *Methods* section. By default no adaptive factors are used.

The SPLITPOINT parameter can be used to save details of the best split found for each X variable. When X is a variate or a factor with ordered levels, this returns a scalar containing the boundary between the two splits. Alternatively, when X is a factor with unordered levels, it returns a variate containing the levels allocated to the first split. The CRITERIONVALUE parameter saves the value of the selection criterion for the best split found for each X variable.

The SELECTED option can be set to a dummy to store the identifier of the X variate or factor used in the best split, and the MAXSPLITPOINT option can save details of the best split, similarly to the SPLITPOINT parameter. The MAXCRITERION option saves the maximum value obtained for the selection criterion, and the NOSELECTION saves a scalar containing the value 0 if a split could be selected or 1 if no further splitting was possible. You can save a logical expression representing the best split using the TESTSPLIT option. So, for example, you can put

```
BASSESS [Y=Yvar; TESTSPLIT=Test; ...]
RESTRICT Yvar; #Test == 1
PRINT Yvar
```

to print the y-values of the individuals in the first successor set. BASSESS takes account of restrictions on Y or on any of the X variates or factors. So you also could now use BASSESS to find the best split on that set.

The WEIGHTS option can supply a variate of weights for the observations. This could be used to supply prior probabilities, or to emphasize units that are perceived as being especially important.

Finally, the TOLERANCE option can be used to modify the tolerance multiplier used internally for example to check for equality of x-values. By default this is set automatically to a value appropriate for the Genstat implementation concerned.

Options: Y, SELECTED, TESTSPLIT, MAXSPLITPOINT, MAXCRITERION, NOSELECTION, FMETHOD, ANTIENDCUTFACTOR, WEIGHTS, TOLERANCE.
Parameters: X, ORDERED, SPLITPOINT, CRITERIONVALUE.

Method

Further general information about classification and regression trees can be found in Breiman *et al*. (1984). The methods used by BASSESS for classification trees are based on Taylor & Silverman (1993). The Gini setting of the FMETHOD option uses the change in Gini information:

$$G = (1 - \sum_k \alpha_k^2) - (\sum_k \beta_{1k}) \times (1 - \sum_k \beta_{1k}^2) - (\sum_k \beta_{2k}) \times (1 - \sum_k \beta_{2k}^2)$$

where α_k is the proportion of individuals in the original set that are in group k, and β_{ik} is the proportion of individuals in successor set i ($i = 1$ or 2) that are in group k. The aim here is to split the individuals into sets to maximize differences between the within-set group probabilities. An equivalent formula (Taylor & Silverman 1993, Section 4) is

$$G = (p_1 \times p_2) \times \{ \sum_k \beta_{1k}^2 + \sum_k \beta_{2k}^2 - \sum_k (\beta_{1k} \times \beta_{2k}) \}$$

where $p_i = \sum_k \beta_{ik}$. The alternative MPI (*mean posterior improvement*) criterion concentrates more on making the group probabilities differ between the successor sets:

$$MPI = (p_1 \times p_2) \times \{ 1 - \sum_k ((\beta_{1k} \times \beta_{2k}) / (\beta_{1k} + \beta_{2k})) \}$$

Taylor & Silverman (1993) note that the term ($p_1 \times p_2$) aims to generate successor sets of similar size, and refer to it as the *anti-end-cut factor* because it aims to avoid sets being produced with only a small number of individuals. They suggest that this should vary according to the complexity of the problem, and instead become

$$\min \{ p_1 \times p_2, p_{low} \times (1 - p_{low}) \}$$

where p_{low} is the reciprocal of the number of groups in the initial set for the `classnumber` setting of the `ANTIENDCUTFACTOR` option, and

$$\min \{ 0.5, 1 / (\sum_k \alpha_k^2) \}$$

for the `reciprocalentropy` setting. The idea is to encourage splits that lead to terminal modes – and to take accounts of the fact that these are more likely to be generated as the number of groups becomes small.

Action with `RESTRICT`

You can request that `BASSESS` operate on only a subset of the units by applying a restriction to the `Y` variate or factor, or to any of the `X` variates or factors, or to the `WEIGHTS` variate.

References

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, Monterey.

Taylor, P.C. & Silverman, B.W. (1993). Block diagrams and splitting criteria for classification trees. *Statistics and Computing*, **3**, 147-161.

See also

Directives: `BCUT`, `BGROW`, `BIDENTIFY`, `BJOIN`, `TREE`.

Procedures: `BCONSTRUCT`, `BCLASSIFICATION`, `BGRAPH`, `BKEY`, `BPRINT`, `BPRUNE`.

Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

BCUT

Cuts a tree at a defined node, discarding the nodes and information below it.

Option

RENUMBER = *string token*	Whether or not to renumber the nodes of the tree (yes, no); default no

Parameters

TREE = *trees*	Trees to be cut
NODE = *scalars*	Node at which to cut each tree
NEWTREE = *trees*	New trees with the information cut; if unspecified, the new tree replaces the original tree
CUTTREE = *trees*	Tree formed from the branches cut from the original tree
OLDNODES = *variates*	Mapping from old nodes to node numbers in a renumbered new tree (as positive numbers) or to nodes in the CUTTREE (as negative numbers)
NEWNODES = *variates*	Mapping from new node numbers in a renumbered tree to the original nodes
CUTNODES = *variates*	Mapping from node numbers in the CUTTREE tree to the original nodes

Description

BCUT provides the basic tree utility of removing an unwanted branch, which is used for example by the BPRUNE procedure. Other tree utilities are described in the description of the TREE directive (which declares and initializes a tree).

The tree to be cut is specified by the TREE parameter, and the NODE parameter indicates the node at which the cut is to be made. The NEWTREE parameter can supply the identifier for the new tree (after removing all the nodes below NODE); if this is not specified, the new tree replaces the original tree. The subtree below NODE can also be saved (as a tree in its own right, with NODE as the root) using the CUTTREE parameter.

The OLDNODES parameter can save a variate containing a mapping from the old nodes to the new nodes. If the node is a member of the new tree the variate contains the number of that node in the NEWTREE, while if it is one of the nodes that are deleted the variate contains −1 multiplied by its number in the CUTTREE. As far as OLDNODES is concerned NODE is regarded as a member of the NEWTREE.

The NEWNODES parameter can save a variate containing the converse mapping from the NEWTREE to the original tree. There is an element for each new node, containing the number of the equivalent node in the original tree. Similarly, the CUTNODES parameter can save a mapping from the CUTTREE to the original tree.

Option: RENUMBER.
Parameters: TREE, NODE, NEWTREE, CUTTREE, OLDNODES, NEWNODES, CUTNODES.

See also

Directives: BCUT, BGROW, BJOIN, TREE.
Procedures: BCONSTRUCT, BCLASSIFICATION, BGRAPH, BKEY, BPRINT, BPRUNE.
Functions: BBELOW, BBRANCHES, BDEPTH, BMAXNODE, BNBRANCHES, BNEXT, BNNODES, BPATH, BPREVIOUS, BSCAN, BTERMINAL.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

BGROW

Adds new branches to a node of a tree.

No options

Parameters

TREE = *trees*	Trees to be extended
NODE = *scalars*	Node at which to extend each tree
NBRANCHES = *scalars*	Number of branches to add to each node; default 2
POSITION = *scalars*	Position at which to add the branches to each node; default * i.e. after all the current braches from the node
NEWNODES = *variates*	Returns the number(s) allocated to the new nodes

Description

BGROW provides the basic tree utility of adding new branches at a node, which is used for example by the BCONSTRUCT procedure. Other tree utilities are described in the description of the TREE directive (which declares and initializes a tree).

The tree to be extended is specified by the TREE parameter, and the NODE parameter indicates the node at which the new branches are to be added. The NBRANCHES parameter specifies the number of branches to add. The POSITION specifies where to add them if the node is a non-terminal node; by default they are added after all the branches currently from the node. The NEWNODES parameter saves a variate containing the numbers of the new nodes (i.e. the terminal nodes at the ends of the new branches).

Options: none.
Parameters: TREE, NODE, NBRANCHES, POSITION, NEWNODES.

See also

Directives: BASSESS, BCUT, BJOIN, TREE.
Procedures: BCONSTRUCT, BCLASSIFICATION, BGRAPH, BKEY, BPRINT, BPRUNE.
Functions: BBELOW, BBRANCHES, BDEPTH, BMAXNODE, BNBRANCHES, BNEXT, BNNODES, BPATH, BPREVIOUS, BSCAN, BTERMINAL.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

BIDENTIFY

Identifies specimens using a tree.

Options

TREE = *tree*	Specifies the tree
TESTELEMENT = *scalar*	Specifies which element of the pointer of information stored at each node of the tree contains the test to be done there to determine which subsequent branch to take
TERMINALNODES = *scalar*, *variate* or *pointer*	Scalar or variate saving the number or numbers of the terminal nodes reached by a single specimen, or pointer of scalars or variates saving the numbers of the terminal nodes reached by several specimens

Parameters

X = *factors* or *variates*	Variables involved in the tests performed in the tree
VALUES = *scalars*, *variates* or *texts*	Values of the variables for the specimens to be identified

Description

BIDENTIFY identifies specimens using a classification tree, or a regression tree, or an identification key (as constructed by procedures BCLASSIFICATION, BREGRESSION or BKEY, respectively).

The characteristics of the specimens are specified using the X and VALUES parameters. Each X setting should be one of the factors or variates in the tree, and the corresponding VALUES setting should be a scalar, variate or text defining its values for the specimens. If X is a variate, VALUES may be a scalar if all the specimens have the same x-value (or if there is only one specimen); it will be a variate if there are several specimens with different x-values. VALUES can be also be a scalar or variate if X is a factor. Alternatively, VALUES may be a text (with one or several values) if the factor X has labels.

The tree is supplied by the TREE option. The TESTELEMENT option indicates which element of the pointer of information, stored at each node of the tree, contains the test to be done there. For trees constructed by procedures BCLASSIFICATION, BREGRESSION or BKEY the test element is the second element of the pointers. In trees constructed by BKEY the test is a factor whose (ordinal) level number defines the branch to take from the node. Alternatively, the tests in trees constructed by BCLASSIFICATION and BREGRESSION are simple logical expressions like

```
X < 1
```

or

```
X .IN. !t(red,blue)
```

where a "true" result selects the first branch, and a "false" result selects the second. BIDENTIFY allows for expressions containing a single relational operator from the following list:

equality	.EQ. or ==
string equality	.EQS.
non-equality	.NE. or /= or <>
string non-equality	.NES.
less than	.LT. or <
less than or equals	.LE. or <=
greater than	.GT. or >
greater than or equals	.GE. or >=
inclusion	.IN.

non-inclusion .NI.

If the factor or variate in the test is not in the list supplied by the X parameter, all the branches from the node must be followed, and the specimen will reach several terminal nodes. All the branches must also be taken if the specimen has a missing value for the X variable in the test.

The TERMINALNODES option saves the numbers of the terminal nodes that the specimens reach in the tree. If there is a single specimen, TERMINALNODES will be a scalar or a variate. If there are several specimens, it will be a pointer of scalars or variates.

Options: TREE, TESTELEMENT, TERMINALNODES.
Parameters: X, VALUES.

Action with RESTRICT
Any restrictions are ignored.

See also
Directives: BASSESS, BCUT, BGROW, BJOIN, TREE.
Procedures: BCONSTRUCT, BCLASSIFICATION, BGRAPH, BKEY, BPRINT, BPRUNE.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

BJOIN

Extends a tree by joining another tree to a terminal node.

No options

Parameters

TREE = *trees*	Trees to be extended
NODE = *scalars*	Node at which to join the tree
JOINTREE = *trees*	Trees to be joined onto the tree
NEWNODES = *variates*	New node numbers allocated to each node in JOINTREE in the new tree

Description

BJOIN provides the basic tree utility of joining a tree to the terminal node of a tree. Other tree utilities are described in the description of the TREE directive (which declares and initializes a tree).

The tree to be extended is specified by the TREE parameter, and the NODE parameter indicates the node at which the tree is to be joined. The JOINTREE parameter specifies the tree to be joined onto the tree, and the NEWNODES parameter saves a variate containing the numbers of the nodes of the JOINTREE in the new tree.

Options: none.
Parameters: TREE, NODE, JOINTREE, NEWNODES.

See also

Directives: BASSESS, BCUT, BGROW, TREE.
Procedures: BCONSTRUCT, BCLASSIFICATION, BGRAPH, BKEY, BPRINT, BPRUNE.
Functions: BBELOW, BBRANCHES, BDEPTH, BMAXNODE, BNBRANCHES, BNEXT, BNNODES, BPATH, BPREVIOUS, BSCAN, BTERMINAL.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

BLOCKSTRUCTURE

Defines the blocking structure of the design and hence the strata and the error terms.

No options

Parameter

formula	Block model (defines the strata or error terms for subsequent ANOVA statements)

Description

The BLOCKSTRUCTURE directive specifies the underlying (or *blocking*) structure of a design that is to be analysed by ANOVA. However, this can be omitted for unstructured designs with a single error term.

In many designs, the units are nested. The simplest is the randomized block design. Here the units are grouped into sets, known as *blocks*, the aim being that units in the same block should be more similar than those in different blocks. The allocation of the treatments is randomized independently within each block. The design thus has two sources of random variation: differences between blocks as a whole, and differences between the units within each block. For example if the units are plots of land and the blocks are groupings of nearby plots we would have two factors: Blocks to indicate the block to which each plot belonged, and Plot to identify the plots within each block. The block model would then be

```
Blocks/Plots
```

indicating that the plots are nested within blocks, and thus that there is no special similarity, for example, between the plot numbered 3 in block 1 and plot 3 of the other blocks. The formula is expanded by Genstat to become

```
Blocks + Blocks.Plots
```

giving terms for the differences between blocks as a whole, and the differences between the units within each block, as required.

In the simplest form of the randomized block design, there is a single treatment factor, each of whose levels occurs once in every block. More complicated arrangements are possible, but each treatment combination must still occur exactly the same number of times in every block. This means that any differences found between the blocks cannot be caused by differences between treatments. Thus the treatment terms are all estimated between the plots within the blocks. If the blocks have been chosen successfully, the variation within the blocks should be less than that between blocks, and so the treatment estimates will be less variable than if a completely randomized design had been used. The analysis of variance will be split into two components called *strata*. The Blocks stratum will contain the sums of squares between blocks; this all arises from the variability between the blocks. The Blocks.Plots stratum will contain the sum of squares for the plots within the blocks; this is partitioned into the sums of squares due to each of the treatment terms, and a residual against which these can be assessed.

Thus, you can deduce the block model from the structure of the units, which should correspond to the way in which the randomization has been done. Genstat expands the block model to form the list of *block* (or *error*) terms, each of which defines a stratum corresponding to one of the sources of variability in the design. Alternatively, if you prefer to deduce the error terms by some other means, as for example if you follow the philosophy of fixed and random effects, you can specify the block model to be the sum of these terms.

In the analysis, Genstat initially partitions the sums of squares according to the block model alone. This gives the total sum of squares for each of the strata. Then it partitions each stratum sum of squares into sums of squares for those treatment terms estimated in that stratum, and a residual which provides an estimate of variability against which these treatment sums of squares should be compared.

In the randomized block design, the treatments are estimated only in the final (bottom) stratum. You would thus get the same sums of squares if you omitted the `BLOCKSTRUCTURE` statement and put Blocks at the start of the treatment model. However the use of `BLOCKSTRUCTURE` better reflects the structure of the design, as it correctly identifies Blocks as an error term. It also allows for the possibility of treatments being estimated between blocks, as in a balanced incomplete-blocks design.

The simplest design in which the treatments are not all estimated in one stratum is the split-plot design. This again has a nested structure and was devised originally for agricultural experiments where some of the factors can be applied to smaller plots of land than others. However, it also occurs in industrial experiments, in medical experiments and even in the study of cake mixtures. An example is shown in Section 6 of the *Genstat for Windows Introduction*. Here there are two treatment factors: three different varieties of oats, and four levels of nitrogen. Because of limitations on the machines for sowing seed, different varieties cannot conveniently be applied to plots as small as those that can be used for the different rates of fertilizer. So the design was set up in two stages. First of all, the blocks were each divided into three plots of the size required for the varieties, and the three varieties were randomly allocated to the plots within each block (exactly as in the randomized blocks design). Then each of these plots, or *whole-plots* as they are usually known, was split into four *sub-plots* (one for each rate of nitrogen), and the allocation of nitrogen was randomized independently within each whole-plot. The design has sub-plots nested within whole-plots, which are themselves nested within the blocks: that is,

```
BLOCKSTRUCTURE Blocks / Wplots / Subplots
```

This expands to

```
Blocks + Blocks.Wplots + Blocks.Wplots.Subplots
```

giving strata for variation between blocks, between whole-plots within the blocks, and for sub-plots within the whole-plots (within blocks). Just as in the randomized block design, the blocks all contain the same sets of treatments, and so no treatments are estimated in the `Blocks` stratum. But varieties, which were applied to whole-plots, are estimated in the `Blocks.Wplots` stratum; in conventional terminology this is called the stratum for whole-plots within blocks. The variance ratio for varieties is calculated by dividing the `Variety` mean square by the `Blocks.Wplots` residual mean square. It is easy to see that this is the correct thing to do. When we look to see whether the varieties differ we are really trying to answer the question: "Do the yields from the three sets of whole-plots, on the first of which the variety Victory was grown, on the second Golden rain and on the third Marvellous, differ by more than the amount that we would expect for any three randomly chosen sets of whole-plots?". Technically, variety is said to be *confounded* with whole plots. The terms for `Nitrogen`, which was applied to sub-plots, and for the `Variety.Nitrogen` interaction are both estimated in the stratum for sub-plots within whole-plots (`Blocks.Wplots.Subplots`).

Because Genstat knows the structure of the design it is thus able to present appropriate variance ratios for the treatment terms. It is also able, for example, to produce correct standard errors and LSDs for tables of means.

There are some designs where the units have a crossed instead of a nested structure. A simple example is the Latin square. This was devised for agricultural experiments to cater for situations where there are fertility trends both along and across the field, but it can be used whenever there are two independent ways of grouping the units: for example time of testing and batch of material, or the litter of the rat and its order by weight within the litter. In field experiments, the plots are arranged in a square, with blocking factors called `Rows` and `Columns`. These each have the same number of levels as there are treatments. Values of the single treatment factor are arranged so that each level occurs once in each row and once in each column. The block structure has rows crossed with columns: that is,

```
BLOCKSTRUCTURE Rows*Columns
```

which expands to

```
Rows + Columns + Rows.Columns
```

The treatments are estimated only in the `Rows.Columns` stratum. Removing variation between rows and between columns should make these estimates less variable.

More complicated designs may involve both crossing and nesting. For example nested Latin squares have the structure

```
BLOCKSTRUCTURE Squares / (Rows * Columns)
```

which gives strata for squares, rows within squares, columns within squares and rows.columns within squares:

```
Squares +Squares.Rows +Squares.Columns +Squares.Rows.Columns
```

Alternatively, a Latin square with split plots for which the structure is defined by

```
BLOCKSTRUCTURE  (Rows * Colums) / Subplots
```

giving the strata of an ordinary Latin square plus an additional stratum for subplots within rows and columns:

```
Rows + Colums + Rows.Colums + Rows.Colums.Subplots
```

If the factors in the block formula do not provide a unique index for every unit of the experiment, the terms in the block model will not account for all the variation. Genstat must then define a final stratum to contain the variation between the sets of units whose levels are the same for each block factor. At the end of the block model, Genstat therefore sets up an extra term containing all the block factors, together with an extra "factor", denoted `*units*`, which numbers the units within each set. So, for the randomized block design, you could put just

```
BLOCKSTRUCTURE Blocks
```

which would then become

```
BLOCKSTRUCTURE Blocks + Blocks.*units*
```

Likewise, for the split-plot design,

```
BLOCKSTRUCTURE Blocks/Wplots
```

would become

```
BLOCKSTRUCTURE Blocks/Wplots + Blocks.Wplots.*units*
```

Consequently, if you define no block structure at all, Genstat assumes

```
BLOCKSTRUCTURE *units*
```

giving a single source of variation representing random differences between the units (this defines a completely randomized design). However, you may prefer to define a more meaningful labelling of the units, for example

```
BLOCKSTRUCTURE Unitcode
```

The factor `Unitcode` would be very easy to set up. To produce a factor equivalent to `*units*` in more complicated situations, you can use procedure `AFUNITS`. For example

```
AFUNITS [BLOCKSTRUCTURE=Blocks/Wplots] Splot
```

to generate a factor `Splots` to index the units within `Blocks` and `Wplots`.

Options: none.
Parameter: unnamed.

See also

Directives: `ANOVA`, `COVARIATE`, `TREATMENTSTRUCTURE`, `ADISPLAY`, `AKEEP`.
Procedures: `AFCOVARIATES`, `ASTATUS`, `AUNBALANCED`.
Genstat Reference Manual 1 Summary section on: Analysis of variance.

BREAK

Suspends execution of the statements in the current channel or control structure and takes subsequent statements from the channel specified.

Option

CHANNEL = *scalar*	Channel number; default 1

Parameter

expression	Logical expression controlling whether or not the break takes place

Description

The BREAK directive allows you to halt the execution of the current set of statements temporarily so that you can execute some other statements. If the parameter is not set, the break will always take place. Alternatively, you can specify a logical expression and then the break will take place only if this produces a *true* (i.e. non-zero) result.

The CHANNEL option determines where the statements to be executed during the break are to be found. Usually (and by default) they are in channel 1. The statements are read and executed, one at a time, until an ENDBREAK statement is reached, at which point control returns to the statements originally being executed.

BREAK also provides a convenient way of interrupting a loop or a procedure so that you can read one set of output before the next is produced.

Option: CHANNEL.
Parameter: unnamed.

See also

Directives: ENDBREAK, DEBUG, CALCULATE.
Genstat Reference Manual 1 Summary section on: Program control.

CALCULATE

Calculates numerical values for data structures.

Options

`PRINT` = *string token*	Printed output required (`summary`); default `*` i.e. no printing
`ZDZ` = *string token*	Value to be given to zero divided by zero (`missing`, `zero`); default `miss`
`TOLERANCE` = *scalar*	If the scalar is non missing, this defines the smallest non-zero number; otherwise it accesses the default value, which is defined automatically for the computer concerned
`SEED` = *scalar*	Seed to use for any random number generation during the calculation; default 0
`INDEX` = *scalar*	If the calculation has a list of structures before the assignment operator (=), the scalar indicates the position within the list of the structure currently being evaluated
`RESTRICTEDUNITS` = *variate*	Defines a "restriction" on the vectors in the expression; if this is set the calculations on those vectors will take place only on the units listed in the variate (and any restrictions of their own will be ignored)

Parameter

expression	Expression defining the calculations to be performed

Description

The `CALCULATE` directive allows you to perform transformations and other calculations. It has the form:

```
CALCULATE expression
```

The *expression* specifies what calculation is to be done, and where the results are to be stored. For example, the command

```
CALCULATE Area = Length * Breadth
```

specifies that the structure `Area` is to store the results of `Length` multiplied by `Breadth`. All the usual arithmetic operators are available:

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponentiation (for example, `X**2` stands for X^2)

`CALCULATE` can operate on any numerical data structure and it will automatically declare the structure to hold the results if you have not declared it already. So, if `Area` has not yet been defined and `Length` and `Breadth` are scalars, `Area` will become a scalar too.

Generally the structures involved in the calculation must have the same "shape" (for example, variates must have the same length) and the operators operate element-by-element over all their values. So, if `Length` and `Breadth` were variates, `Area` would become a variate each of whose units contained the product of the corresponding units of `Length` and `Breadth`. However, scalars and ordinary numbers can be included with calculations on any type of data structure. So

```
CALCULATE Kilo = Pound / 2.2
```

would be valid whatever the type of the structures `Kilo` and `Pound`.

If any of the values involved in a numerical expression is missing, the result will be missing

too.

Genstat has operators for relational tests:

== or .EQ.	equality of numerical values
.EQS.	equality of textual strings
>= or .GE.	greater than or equal to
> or .GT.	greater than
<= or .LE.	less than or equal to
< or .LT.	less than
/= or <> or .NE.	not equal to
.NES.	inequality of textual strings
.IS.	identifier equivalence (to test whether a dummy contains a particular identifier)
.ISNT.	identifier non-equivalence
.IN.	inclusion: X.IN.Vals gives result true for each value of X that is equal to any one of the values of Vals
.NI.	non-inclusion: the opposite of .IN.

These generate a result of zero if the test is false, and one if it is true. (In fact any non-zero value is taken to represent a true value.) With most of these operators, a missing value in either operand (or in both) will generate a missing result. The exceptions are .EQ. and .NE. (and their synonyms), and .EQS. and .NES.: when both operands are missing .EQ. and .EQS. give a true result, while .NE. and .NES. give a false result.

There are also logical operators that can be used to combine the results of expressions involving relational operators.

.AND.	and: a.AND.b true if both a and b are true
.EOR.	either or: a.EOR.b is true if either a or b, but not both, is true
.OR.	or: a.OR.b is true if either a or b is true
.NOT.	not: .NOT.a is true for a untrue

The precedence rules of the operators are very similar (but possibly not identical) to those in computer languages like C or Fortran. The list below shows the order in which the operators are evaluated when they are used in expressions, if brackets are not used to make the order explicit:

1) .NOT. Monadic -
2) .IS. .ISNT. .IN. .NI. *+
3) **
4) * /
5) + Dyadic -
6) < > == <= >= /= <> .LT. .GT. .EQ. .LE. .GE. .NE. .NES.
7) .AND. .OR. .EOR.
8) =

(Monadic minus means the use of the minus sign in a negative number: for example, -1.) Within each class, operations are done from left to right within an expression, unless brackets are used to indicate some other order. So

```
A > B+C/D*E
```

is the same as

```
A > ( B + ( (C/D) * E )
```

Expressions can contain lists, to specify that the same calculation is to be done for several sets of structures. For example

```
CALCULATE Pay1,Pay2 = Hours1,Hours2 * Rate + Bonus
```

This has the same effect as the two commands

```
CALCULATE Pay1 = Hours1 * Rate + Bonus
CALCULATE Pay2 = Hours2 * Rate + Bonus
```

Notice that, if any of the lists on the right-hand side of the expression is shorter than the list on the left-hand side, the list is re-used. So the value of `Bonus` is used for both calculations. To take a more complicated example

```
CALCULATE X,Y,Z = A,B,C + 1,2
```

is the same as the three calculations

```
CALCULATE X = A + 1
CALCULATE Y = B + 2
CALCULATE Z = C + 1
```

However, the lists on the right-hand side must not be longer than the list on the left-hand side.

When the calculation contains lists, you can set the `INDEX` option to a scalar which will contain the index of the current calculation. For example

```
CALCULATE [INDEX=i] X,Y,X = i * A,B,C
```

is the same as the three calculations

```
CALCULATE X = 1 * A
CALCULATE Y = 2 * B
CALCULATE Z = 3 * C
```

as `X` and `A` are the first items of their lists, `Y` and `B` are the second, and `Z` and `C` are the third.

Genstat provides a wide range of functions for use in expressions. Many of these, known as transformations, produce a result that is the same type of structure as the *argument* of the function. For example,

```
CALCULATE Logsulph = LOG(Sulphur)
```

uses the `LOG` function to take natural logarithms of the values in the data structure `Sulphur`. If `Sulphur` is a variate `Logsulph` will also be a variate with the same number of values.

Scalar functions produce a scalar summary of all the values in a structure. For example, we can use the `SUM` function to calculate the total `Sulphur` values:

```
CALCULATE Totsulph = SUM(Sulphur)
```

There are also vector functions that produce summaries across the values of a set of variates (or of scalars). The set of variates must be put into a pointer. So, we could form a variate `M` each of whose units contains the mean of the values in the corresponding units of the variates `A`, `B` and `C` by

```
POINTER [VALUES=A,B,C] Vars
CALCULATE M = VMEAN(Vars)
```

This can be done more succinctly using an unnamed pointer:

```
CALCULATE M = VMEAN(!p(A,B,C))
```

When a function has more than one argument, each is separated from the next by a semicolon. For example

```
CALCULATE Corr = CORRELATION(X; Y)
```

calculates the correlation between the values in `X` and `Y`.

Function arguments can also be lists, running in parallel with the other lists in the expression. For example, to calculate `Corr1` as the correlation between `X1` and `Y1`, and `Cor2` as the correlation between `X2` and `Y2`:

```
CALCULATE Corr1,Corr2 = CORRELATION(X1,X2; Y1,Y2)
```

When a factor occurs in an expression on the right-hand side, Genstat usually works with its levels. The exception is when the factor occurs as the first operand of the operators `.IN.` or `.NI.` and the second operand is a text; the factor labels are then used instead. A factor can also

occur on the left-hand side of an expression and receive the results of a calculation; an error is reported if any of the resulting values is not one of the levels of the factor. Two functions are provided especially for factors: `NLEVELS(F)` gives the number of levels of the factor `F`, and `NEWLEVELS(F; V)` forms a variate from the factor `F`, using variate `V` to define values for the levels.

Text structures are allowed only with the relational operators `.EQS.`, `.NES.`, `.IN.` and `.NI.` or in the string functions. The result of any expression is a number, so you cannot create a text with `CALCULATE`, even if the structures on which the operations are being done are texts.

All the arithmetic, relational and logical operators and transformation functions can also be used with matrix structures, symmetric matrices and diagonal matrices. The basic rule when using these with different types of matrix is that their dimensions must conform. This means that, for each pair of matrices, row dimension must match row dimension, and column dimension must match column dimension. So, for example, you can add a diagonal matrix to a matrix structure provided the number of rows and columns of the matrix equals the number of rows (and columns) of the diagonal matrix. The multiplication operator (`*`) performs element-by-element multiplication of two matrices: for matrix multiplication, there is the compound operator *+ or the function `PRODUCT`, which is one of the many specialised matrix functions.

You can use tables in expressions in much the same way as you would any other numerical structure. Tables in expressions must be either all without margins or all with margins. If you try to mix tables with and without margins, Genstat will report an error. Calculations with tables are very straightforward when they have the same factors in their classifying sets. The tables then have identical "shapes", and the arithmetic, relational, and logical operators and the transformation functions act element-by-element, in the usual way. When tables have different classifying sets, there are two cases to consider. The first case is when the table on the left-hand side has a factor in its classifying set that is not in the classifying set of the table on the right-hand side. In this case, the right-hand table is expanded to include that factor, by duplicating its values across the levels of the factor and any margin. The second case is when the table on the right-hand side has a factor in its classifying set that is not in the classifying set of the table on the left-hand side. Now the values in the margin over that factor are taken for the left-hand table. If the table has no margins, they must be calculated first. By default Genstat forms marginal totals, but you can use the special table functions to form other types of margin.

Dummies can be used with the relational operators `.IS.` and `.ISNT.` which test whether or not a dummy points to a particular identifier. For example, to store in `Sca` the result of a test to check whether dummy `D` points to `Va`, you would put

```
CALCULATE Sca = D.IS.Va
```

while to test that `D` does not point to `Vb`, you would put

```
CALCULATE Sca = D.ISNT.Vb
```

There are also the functions `SET` and `UNSET` to test if a dummy has or has not been set to any value. Other specialised functions include subset functions, statistical functions and random number generation functions.

`CALCULATE` has four options: `PRINT`, `ZDZ`, `TOLERANCE` and `SEED`. If you set the `PRINT` option to `summary`, Genstat will print some summary information every time that values are assigned to a structure. The information has the same form as in the `READ` directive: identifier, minimum value, mean value, maximum value, number of values, number of missing values, and whether or not the set of values is skew.

If you try to use `CALCULATE` to do something invalid, such as the logarithm or the square root of a negative number, Genstat generates a warning diagnostic and inserts a missing value in the offending unit. The one exception is the division of zero by zero, which is regarded as deliberate. Genstat thus does not print a diagnostic, but uses option `ZDZ` to determine whether the result should be a missing value (`ZDZ=missing`) or zero (`ZDZ=zero`); the default is `missing`.

The `SEED` option provides the seed to generate random numbers for the functions `GRBETA`,

GRBINOMIAL, GRCHISQUARE, GRF, GRGAMMA, GRHYPERGEOMETRIC, GRLOGNORMAL, GRNORMAL, GRPOISSON, GRT and GRUNIFORM if these occur in the expression. The seed can be any non-negative integer, but only the last six digits of its integer part are used. Thus the seeds 2144556 and 7144556.3 are both equivalent to the seed 144556. The default value of zero continues an existing sequence of random numbers, if either these functions or the function URAND (which has its own argument to set the seed) has already been used in the current Genstat run. If, however, this is the first time that these functions have been used, Genstat picks a random seed.

The RESTRICTEDUNITS option allows you to apply a "restriction" to the vectors in the expression. Its setting is a variate containing a list of the units numbers on which you want the calculation to be done (the other units are then ignored). This works in the same way as if you had applied a restriction on one of these vectors explicitly, using the RESTRICT directive (see below). However, if RESTRICTEDUNITS is set, restrictions on the vectors themselves are ignored. By default, when RESTRICTEDUNITS is unset, CALCULATE will look for restrictions in the vectors, as usual. Note, though, that you can set RESTRICTEDUNITS=* to make the calculation work on all the units, regardless of whether any of the vectors is restricted.

Options: PRINT, ZDZ, TOLERANCE, SEED, INDEX.
Parameter: unnamed.

Action with RESTRICT
If you are calculating values for a variate or factor, you can restrict the operation to only a subset of the units by applying a restriction to any of the variates, factors or texts involved in that calculation. The values in the other units are left unchanged. If more than one of these vectors is restricted, they must all be restricted in the same way. Note, though, that restrictions on a variate within a scalar function (for example MEAN), or within the RESTRICTION function, operate independently from the main calculation outside. Also, restrictions in the main calculation are ignored if it contains qualified identifiers or the ELEMENTS function.

See also
Directives: EXPRESSION, SETCALCULATE, NAG, FLRV, QRD, SVD.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

CALLS

Lists library procedures called by a procedure.

No options

Parameter

identifiers	Names of the called procedures

Description

CALLS is useful when you are building a suite of procedures. By default, when you define a procedure, Genstat checks that any procedures that it calls are available in the program or in an attached procedure library. However, this can create problems when procedures call each other. For example, Genstat is happy to execute programs where a procedure, A say, calls other procedures that themselves call procedure A. However, it is then impossible to decide on an order in which to define the procedures.

Before CALLS became available, the solution was to define dummy procedures (with option and parameter definitions but no executable statements) before any of the real procedures were defined. A better solution now, though, is to specify a CALLS statement in each procedure, to list the procedure that it calls. Genstat then regards these as a set of "trusted" sub-procedures, that it assumes will be provided before the procedure is executed. (If not, you will get a fault diagnostic then!)

The CALLS statement must come immediately after the option and parameter definitions (using the OPTION and PARAMETER directives), and before any executable statements. It has a single parameter, that lists the names of the procedures that are called.

Options: none.
Parameter: unnamed.

See also

Directives: PROCEDURE, OPTION, PARAMETER, ENDPROCEDURE.
Genstat Reference Manual 1 Summary section on: Program control.

CAPTION

Prints captions in standardized formats.

Option

PFIRST = *string tokens*	What to print first (dots, page, outprint); default * i.e. none

Parameters

TEXT = *texts*	Contents of the captions
STYLE = *string tokens*	Style for each caption (plaintext, stress, minor, major, meta, note, status); default plai

Description

The CAPTION directive allows various types of caption to be printed in the standard Genstat styles. The contents of the caption are supplied by the TEXT parameter. The STYLE parameter specifies a string to indicate the caption style:

plaintext	ordinary text,
stress	text to be emphasized,
minor	a minor caption signifying a sub-section in the output,
major	a major caption signifying a section in the output,
meta	a meta-caption to group several sections of output,
note	a "note" to the user, and
status	a "status" message.

The PFIRST option allows you to start the caption on a new page or to precede it by a line of dots (or a horizontal "rule" if the output is formatted; see the OPEN directive). Alternatively, the outprint setting generates the dots or new page according to the setting for the current output channel (see the OUTPUT directive).

Option: PFIRST.
Parameters: TEXT, STYLE.

See also

Directives: PAGE, PRINT, SKIP, TXCONSTRUCT.
Genstat Reference Manual 1 Summary section on: Input and output.

CASE

Introduces a "multiple-selection" control structure.

No options

Parameter

expression	Expression which is evaluated to an integer, indicating which set of statements to execute

Description

A *multiple-selection* control structure consists of several alternative blocks of statements. The first of these is introduced by a `CASE` statement. This has a single parameter, which is an expression that must yield a single number. Subsequent blocks are each introduced by an `OR` statement. There can then be a final block, introduced by an `ELSE` statement. The whole structure is terminated by an `ENDCASE` statement. Thus the general form is: first

`CASE` expression
  statements

then either none, one or several blocks of statements of the form

`OR`
  statements

then, if required, a block of the form

`ELSE`
  statements

and finally the statement

`ENDCASE`

Genstat rounds the expression in the `CASE` expression to the nearest integer, *k* say, and then executes the *k*th block of statements. If there is no *k*th block present (as for example if *k* is negative) the block of statements following the `ELSE` statement is executed, if there is such a block; otherwise an error diagnostic is given.

This example prints the salient details about each day in the song *The twelve days of Christmas*. The scalar `Day` indicates which day it is.

```
CASE Day
  PRINT 'a partridge in a pear tree'
OR
  PRINT 'two turtle doves and a partridge in a pear tree'
OR
  PRINT 'three French hens, two turtle doves \
    and a partridge in a pear tree'
OR
  PRINT 'four calling birds, three French hens ...'
OR
  PRINT 'five gold rings ...'
OR
  PRINT 'six geese a-laying ...'
OR
  PRINT 'seven swans a-swimming ...'
OR
  PRINT 'eight maids a-milking ...'
OR
  PRINT 'nine drummers drumming ...'
OR
  PRINT 'ten pipers piping ...'
OR
  PRINT 'eleven ladies dancing ...'
```

```
OR
  PRINT 'twelve lords a-leaping ...'
ELSE
  PRINT 'sorry, no delivery today'
ENDCASE
```

`CASE` statements can be nested to any depth.

Options: none.
Parameter: unnamed.

See also
Directives: `OR`, `ELSE`, `ENDCASE`, `EXIT`, `IF`, `FOR`, `CALCULATE`.
Genstat Reference Manual 1 Summary section on: Program control.

CATALOGUE

Displays the contents of a backing-store file.

Options

PRINT = *string tokens*	What to print (subfiles, structures); default subf, stru
CHANNEL = *scalar*	Channel number of the backing-store file; default 0, i.e. the workfile
LIST = *string token*	How to interpret the list of subfiles (inclusive, exclusive, all); default incl
SAVESUBFILE = *text*	To save the subfile identifiers; default *
UNNAMED = *string token*	Whether to list unnamed structures (yes, no); default no

Parameters

SUBFILE = *identifiers*	Identifiers of subfiles in the file to be catalogued
SAVESTRUCTURE = *texts*	To save the identifiers of the structures in each subfile

Description

You can use CATALOGUE to obtain details of the subfiles contained in a backing-store file, or the structures within an ordinary subfile, or the procedures within a procedure subfile. The file is indicated by the CHANNEL option, and the SUBFILE parameter specifies the subfiles (of ordinary structures or of procedures) that are to be catalogued.

The PRINT option specifies which catalogues are to be printed. The subfiles setting prints the catalogue of subfiles in the backing-store file attached to the channel specified by the CHANNEL option, while the structures setting prints the catalogue of structures or procedures that are in the subfiles specified by the SUBFILE parameter. If you set option UNNAMED=yes the unnamed structures in each subfile will also be listed, together with details of how the structures depend on each other.

The LIST option controls how the SUBFILE list is interpreted. The default setting inclusive simply catalogues the subfiles that have been listed. Alternatively, if you set LIST=all Genstat will catalogue all the subfiles in the backing-store file. Finally, you can see LIST=exclusive to catalogue everything that you have not included in the SUBFILE list.

The SAVESTRUCTURE parameter allows you to set up texts, one for each subfile in the SUBFILE parameter. Each text contains the identifiers of all structures with an unsuffixed identifier in the subfile. Each identifier is put on a separate line, and the characters ,\ are appended to all but the last line. You would normally use these texts as a macro; the ,\ makes them useable as lists of identifiers. If the text is used as a macro, it is subject to the restriction on the length of statements. The SAVESUBFILE option allows you to save a similar text containing the identifiers of all the subfiles in a backing-store file.

Options: PRINT, CHANNEL, LIST, SAVESUBFILE, UNNAMED.
Parameters: SUBFILE, SAVESTRUCTURE.

See also

Directives: STORE, RETRIEVE, MERGE.
Genstat Reference Manual 1 Summary section on: Input and output.

CLOSE
Closes files.

No options

Parameters

CHANNEL = *scalars* or *texts*	Numbers of the channels to which the files are attached, or identifiers of texts used for input (which, after "closing", can then be re-read)
FILETYPE = *string tokens*	Type of each file (input, output, unformatted, backingstore, procedurelibrary, graphics); default inpu
DELETE = *string tokens*	Whether to delete the file on closure (yes, no); default no

Description
Once you have finished using a file, CLOSE can be used to release the channel to which it is attached, so that the channel is available for use with some other file. Parameters CHANNEL and FILETYPE indicate the channel number and the type of file, as in the OPEN directive. The DELETE parameter is useful if you are using files to store data temporarily, perhaps to release workspace within Genstat. When you have finished with the file you can set DELETE=yes to request that it be deleted on closure so that disk space is not wasted. For example,

```
OPEN   'temp.bin'; CHANNEL=3; FILETYPE=unformatted
PRINT  [CHANNEL=3;UNFORMATTED=yes] \
       Surveys[1900,1910...1990]
DELETE Surveys[1900,1910...1990]

"... and later on when you wish to retrieve the data ..."
READ   [CHANNEL=3;UNFORMATTED=yes] \
       Surveys[1900,1910...1990]
CLOSE  3; FILETYPE=unformatted; DELETE=yes
```

You cannot close a channel to which the keyboard or screen are attached, nor the current input or output channels. Also you cannot use CLOSE to delete files that have been opened with ACCESS=readonly or that are protected by the computer's file system. However, you do not need to close every file before you stop running Genstat; files are automatically closed at the end of every Genstat program.

Options: none.
Parameters: CHANNEL, FILETYPE, DELETE.

See also
Directives: OPEN, ENQUIRE, FCOPY, FDELETE, FRENAME.
Genstat Reference Manual 1 Summary section on: Input and output.

CLUSTER

Forms a non-hierarchical classification.

Options

PRINT = *string tokens*	Printed output required (criterion, optimum, units, typical, initial, random); default * i.e. no printing
DATA = *matrix* or *pointer*	Data from which the classification is formed, supplied as a units-by-variates matrix or as a pointer containing the variates of the data matrix
CRITERION = *string token*	Criterion for clustering (sums, predictive, within, Mahalanobis); default sums
INTERCHANGE = *string token*	Permitted moves between groups (transfer, swop); default tran (implies swop also)
START = *factor*	Initial classification; default * splits the units, in order, into NGROUPS classes of nearly equal size
NSTARTS = *scalar*	Number of random starting configurations to be used; default 0
SEED = *scalar*	Seed for the random numbers used to form random starting configurations; default 0

Parameters

NGROUPS = *scalars*	Numbers of classes into which the units are to be classified: note, the values of the scalars must be in descending order
GROUPS = *factors*	Saves the classification formed for each number of classes
CRITERIONVALUE = *scalars*	Saves the criterion values (representing within-class homogeneity)
BCRITERIONVALUE = *scalars*	Saves the subsidiary criterion values (representing between-class heterogeneity for maximal predictive classification)
MEANS = *matrices*	Saves the variate means for the groups of each classification
PREDICTORS = *matrices*	Saves the group predictors from maximal predictive classification

Description

Printed output is controlled by the PRINT option. This has the following possible settings.

criterion	prints the optimal criterion value.
optimum	prints the optimal classification.
units	prints the data with the units ordered into the optimal classes.
typical	prints a typical value for each class: for maximal predictive classification this is the class predictor; for the other methods it is the class mean.
initial	if this is set, the requested sections of output are also printed for the initial classification.
random	if this is set, the requested sections of output are also printed for the optimum configuration obtained from every random start.

The DATA option supplies the data to be classified. This specifies a single structure that must

be either a matrix, with rows corresponding to the units and columns to the variables, or a pointer whose values are the identifiers of the variates in the data matrix. Internally, CLUSTER operates on a matrix, and so it will copy the variate values into a matrix if you supply a pointer as input; thus, it is more efficient to supply a matrix, especially with large data sets.

The CRITERION option specifies which criterion CLUSTER is to optimize. The four available settings are:

sums	minimize the within-group sum of squares (and thus maximize the between-group sum of squares);
predictive	maximal predictive classification;
within	minimize the determinant of the pooled within-class dispersion matrix;
mahalanobis	maximize the total Mahalanobis squared distance between the groups.

The default is sums.

The INTERCHANGE option specifies which types of interchange (transfers or swops) are to be used. The default is transfer, which is taken to imply that both transfers and swops are used, since a swop is simply two transfers. If you set INTERCHANGE=swop, only swops are used. If INTERCHANGE=* the algorithm does not attempt to improve the classification from the initial classification; you might want this, in conjunction with the PRINT=initial setting, to display the results for an existing classification which you do not wish to improve.

The START option can be used to supply a factor to define the initial classification. This might be constructed using the CLASSIFY procedure. If there are *k* classes, CLASSIFY finds the *k* units that are furthest apart in the multi-dimensional space defined by the data variates. These are then used as the nuclei for the classes, with each remaining unit being allocated to the class containing the nearest nucleus. The default splits the units, in order, into NGROUPS classes of nearly equal size.

As an alternative to the use of CLASSIFY, the NSTARTS option allows you to specify a number of random permutations of the initial classification to try. CLUSTER then saves the best classification that it finds. By default, NSTARTS=0, i.e. no randomization is done. The SEED option supplies the seed for the random numbers that are used to do the permutations. The default of zero continues the existing sequence of random numbers, if CLUSTER has already been used in the current Genstat job. If CLUSTER has not yet been used, Genstat picks a seed at random.

The first parameter, NGROUPS, specifies the number of groups, or classes, to be formed. Often you would want several classifications from a single data set, into different numbers of groups. In this case, the NGROUPS parameter should be a list of scalars, defining the numbers of groups in descending order. For the initial classification of the second classification, CLUSTER takes the optimal classification from the first number of groups, and does some reallocation of units to make a smaller number of groups. This is repeated, as often as required, to provide initial classifications for all the later analyses; hence the need to specify the numbers in descending order. Random starts are done only for the first number of groups.

The GROUPS parameter can specify a list of factors to save the optimal classifications. The CRITERIONVALUE parameter can specify a list of scalars to save the criterion values for each number of groups. The subsidiary criterion values involved in maximal predictive classification can be saved (also in scalars) using the BCRITERIONVALUE parameter. The MEANS parameter can save matrices containing the means of the variates within the groups of the classifications, and the PREDICTORS parameter can save matrixes containing the group predictors from maximal predictive classifications.

Options: PRINT, DATA, CRITERION, INTERCHANGE, START, NSTARTS, SEED.
Parameters: NGROUPS, GROUPS, CRITERIONVALUE, BCRITERIONVALUE, MEANS, PREDICTORS.

Action with `RESTRICT`
Any restrictions, for example on the variates in a `DATA` pointer, are ignored.

See also
Directives: `FSIMILARITY`, `HCLUSTER`, `HREDUCE`.
Procedures: `CLASSIFY`, `BCLASSIFICATION`, `CINTERACTION`, `HCOMPAREGROUPINGS`, `MASCLUSTER`.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

COKRIGE

Calculates kriged estimates using a model fitted to the sample variograms and cross-variograms of a set of variates.

Options

`PRINT` = *string token*	Controls printed output (`description`, `search`, `weights`, `conditionalprobabilities`, `quantiles`, `crossvalidations`); default `desc`
`Y` = *variate*	Variate to predict in the cokriging
`METHOD` = *string token*	Type of kriging (`Normal`, `LogNormal`); default `Norm`
`X1OUTER` = *variate*	Variate containing 2 values to define the bounds of the region to be examined in the first direction; by default the whole region is used
`X2OUTER` = *variate*	Variate containing 2 values to define the bounds of the region to be examined in the second direction; by default the whole region is used
`X3OUTER` = *variate*	Variate containing 2 values to define the bounds of the region to be examined in the third direction; by default the whole region is used
`X1INNER` = *variate*	Variate containing 2 values to define the bounds of the interpolated region in the first direction; no default
`X2INNER` = *variate*	Variate containing 2 values to define the bounds of the interpolated region in the second direction; no default
`X3INNER` = *variate*	Variate containing 2 values to define the bounds of the interpolated region in the third direction; no default
`X1INTERVAL` = *scalar*	Distance between successive interpolations in the first direction; default 1.0
`X2INTERVAL` = *scalar*	Distance between successive interpolations in the second direction; default 1.0
`X3INTERVAL` = *scalar*	Distance between successive interpolations in the third direction; default 1.0
`POINTS` = *matrix*	Allows the point where predictions are required to be specified explicitly if the `X1-3INNER` and `X1-3INTERVAL` options are unset, otherwise if these are set, saves the locations of the prediction points
`BLOCKDIMENSIONS` = *variate* or *matrix*	Dimensions of the block(s) in the 3 directions, a variate defines identical blocks for each prediction point, a matrix can be used to define different block sizes for each point when the points are defined by the `POINTS` option; default `!(0,0,0)` i.e. punctual kriging at every point
`POOLRADIUS` = *scalar*	Specifies the minimum distance for which points are pooled; default `*` i.e. no pooling
`SEARCHNEIGHBOURHOOD` = *string token*	Search neighbourhood to be used (`global`, `local`); default `glob`
`MINPOINTS` = *scalars*	Minimum number of data points from which to compute elements
`MAXPOINTS` = *scalars*	Maximum number of data points in each direction from which to compute elements

RADII = *scalars* or *variates*	Scalar defining the maximum distance between target point in block and usable data for each variable in 1 dimension, or radii of the ellipse or ellipsoid enclosing the usable points in 2 or 3 dimensions
ELLIPSEAXIS = *scalar* or *variate*	Angle or angles defining the direction of the axis of the ellipse or ellipsoid, scalar for 2 dimensions and variate containing 3 values for 3 dimensions
DRIFT = *string token*	Mean function for universal cokriging (constant, linear, quadratic, polygon); default cons
X1EXV = *variate*	Variate containing locations of the explanatory model in the first dimension
X2EXV = *variate*	Variate containing locations of the explanatory model in the second dimension (if recorded in 2 or 3 dimensions)
X3EXV = *variate*	Variate containing locations of the explanatory model in the third dimension (if recorded in 3 dimensions)
TERMS = *variates*	List of variates for explanatory model; default * i.e. none
POLYGONCOORDINATES = *pointer*	Pointer containing the coordinates of polygons in 2 variates and the map unit numbers within a factor
COORDSYSTEM = *string token*	Coordinate system used for the geometry for discretizing the lag (mathematical, geographical); default math
CPTHRESHOLD = *scalar or variate*	Threshold(s) for calculating the conditional probabilities
PERCENTQUANTILES = *scalar or variate*	Percentage points for which quantiles are required; default 5 and 95
LOGBASE = *string token*	Base of antilog transformation to be applied to the predictions and variances for lognormal (co)kriging (ten, e); default * i.e. none

Parameters

DATA = *variates*	Measurements as one or more variates
X1 = *variates*	Locations of the measurements in the first dimension
X2 = *variates*	Locations of the measurements in the second dimension (if recorded in 2 or 3 dimensions)
X3 = variates	Locations of the measurements in the third dimension (if recorded in 3 dimensions)
PREDICTIONS = *variate*	Kriged estimates
VARIANCES = *variate*	Estimation variances
MEASUREMENTERROR = *scalars*	Variance of measurement error for punctual (co)kriging
ESTIMATES = *pointers*	Estimates for the model structure
CONDITIONALPROBABILITIES = *pointers*	Structure to save conditional probabilities
QUANTILES = *pointers*	Structure to save estimated quantiles
SAMPLESUPPORT = *scalars*	Sampling size (length, area or volume according to the dimensionality of the data) of the data points

Description

The COKRIGE directive computes kriged estimates using a model fitted by MCOVARIOGRAM to the sample auto- and cross-variograms of a set of variates. The data are supplied as a list of variates using the DATA parameter. The target variable to predict is supplied using the Y option. Note that the target variable must also be present in the list of variates supplied with the DATA

parameter. The locations of the measurements are supplied using the parameters X1, X2 (for two or three dimensions) and X3 (for three dimensions).

The METHOD option allows you to specify whether to perform Normal or logNormal cokriging. The lognormal setting is only available for punctual cokriging. For logNormal cokriging the LOGBASE option allows you to specify the base of the logarithms (ten or e) for back transforming the kriged predictions and variances.

By default, Genstat uses global prediction where, for each prediction, all the data values are used. However, it is often desirable to use a subset in a (spatial) neighbourhood around the prediction location. This could be for computational reasons, or to assume local first-order stationarity. You can choose whether to use a global or local search using the SEARCHNEIGHBOURHOOD option.

You can select a subset of the data to be considered when forming the cokriging system by specifying the area or volume defined by X1OUTER, X2OUTER and X3OUTER. Each of these should be set to a variate with two values to define the lower and upper limits in each direction.

You can supply the positions at which the target variable is predicted (estimated) in two ways. The first way is to generate the locations using the X1-3INNER and X1-3INTERVAL options. X1INNER, X2INNER and X3INNER are set to variates with two values to define the lower and upper limits in each direction, and the limits should not lie outside those of X1OUTER, X2OUTER and X3OUTER. X1INTERVAL, X2INTERVAL and X3INTERVAL are set to scalars to define the distance between the successive positions in the first, second and third direction. The intervals should be specified using the same units as the data. You can save the generated locations by supplying an identifier in the POINTS option. The second way is to explicitly supply the points where predictions are required. If the X1-3INNER and X1-3INTERVAL options are unset then you can use the POINTS option to supply a matrix of prediction locations.

By default the cokriging is punctual, i.e. at points that have the same size and shape as the sample support. The BLOCKDIMENSIONS option can be used to specify block cokriging. You can either specify a variate containing the dimensions of the block(s) in the three directions or alternatively supply a matrix defining different block sizes for each point when points are supplied using the POINTS option. For punctual cokriging, you can specify the variance of any measurement error using the MEASUREMENTERROR parameter.

The minimum and maximum number of points used for the kriging are set by the MINPOINTS and MAXPOINTS options, respectively.

The RADII option defines the maximum distance between the target point in a block and usable data. For an isotropic search you should supply a scalar to define the maximum distance or radii of the ellipse (two dimensions) or ellipsoid (three dimensions). For an anisotropic search you should supply the distances for each axis of the ellipse of ellipsoid. For an anisotropic search the angle or angles defining the direction of the axes of the ellipse or ellipsoid for the search are supplied using the ELLIPSEAXIS option. For two dimensions you should supply a scalar containing the angle for the first axis which is measured in degrees, counter-clockwise from East if option COORDSYSTEM is set to mathematical, or clockwise from North if COORDSYSTEM is set to geographical. For three dimensions the first value defines the angle for the first axis which is measured in degrees, counter-clockwise from East if COORDSYSTEM is set to mathematical, or clockwise from North if COORDSYSTEM is set to geographical. The second value defines the dip angle for the first axis (rotation angle around the y-axis) which is measured in degrees up from horizontal. The third value defines the rotation angle of the second and third axis around the first axis (defined by the two previous angles).

The POOLRADIUS option allows you to specify a minimum distance for which points can be pooled.

The ESTIMATES parameter allows you to specify an identifier of a data structure storing estimates of the non-linear parameters, sill values and associated information. This structure should be formed using the MCOVARIOGRAM directive.

The PRINT option controls the printed output with settings:

description	description of the length, area or volume being kriged and the model that is used,
search	the results of the search for data around each position that is kriged,
weights	the kriging weights at each position,
crossvalidation	cross-validation statistics for punctual cokriging (the cross-validation is calculated by estimating each sample point from the data after excluding the sample value),
conditionalprobabilities	conditional probabilities for the values specified by the CPTHRESHOLD option,
quantiles	quantiles for the values specified by the PERCENTQUANTILES option.

Universal kriging may be invoked by setting the DRIFT option to linear or to quadratic, i.e. to be of order 1 or 2. The default is DRIFT=constant, to give ordinary cokriging. You can include explanatory variables in the mean function by listing explanatory variates with the TERMS option, and their associated coordinates using the X1EXV, X2EXV and X3EXV options. For two-dimensional cokriging, the DRIFT=polygon option allows you to specify categorical variables defined by one or more closed polygons (map units). The map units and polygons should be supplied in a pointer using the POLYGONCOORDINATES option. The pointer should contain the coordinates of the polygons in two variates (x- and y-positions) and a factor where each level defines a different map unit. If there is more than one polygon within a map unit these should be separated with a row of missing values.

You can specify the sampling support size (length, area or volume) of the data points using the SAMPLESUPPORT parameter.

The PERCENTQUANTILES option can specify percentage values for which to compute quantiles for the conditional distributions. The quantiles can be saved using the QUANTILES parameter.

The CPTHRESHOLD option allows you to specify thresholds for calculating conditional probabilities. The conditional probabilities can be saved using the CONDITIONALPROBABILITIES parameter.

The kriged predictions and variances can be saved using the PREDICTIONS and VARIANCES parameters. If a grid or volume of points has been generated using the X1-3INNER and X1-3INTERVAL options, the corresponding prediction locations can be saved using the POINTS option.

Options: PRINT, Y, METHOD, X1OUTER, X2OUTER, X3OUTER, X1INNER, X2INNER, X3INNER, X1INTERVAL, X2INTERVAL, X3INTERVAL, POINTS, BLOCKDIMENSIONS, POOLRADIUS, SEARCHNEIGHBOURHOOD, MINPOINTS, MAXPOINTS, RADII, ELLIPSEAXIS, DRIFT, X1EXV, X2EXV, X3EXV, TERMS, POLYGONCOORDINATES, COORDSYSTEM, CPTHRESHOLD, PERCENTQUANTILES, LOGBASE.

Parameters: DATA, X1, X2, X3, PREDICTIONS, VARIANCES, MEASUREMENTERROR, ESTIMATES, CONDITIONALPROBABILITIES, QUANTILES, SAMPLESUPPORT.

See also

Directives: FCOVARIOGRAM, MCOVARIOGRAM, FVARIOGRAM, KRIGE.

Procedures: DCOVARIOGRAM, KCROSSVALIDATION, MVARIOGRAM, DVARIOGRAM, DHSCATTERGRAM.

Genstat Reference Manual 1 Summary section on: Spatial statistics.

COLOUR

Defines the red, green and blue intensities to be used for the Genstat colours for certain graphics devices.

Option

RESET = *string token*	Whether to reset values to their defaults (yes, no); default no

Parameters

NUMBER = *scalars*	Numbers of the colours to be set
RED = *scalars*	Red intensity of each colour (between 0 and 255)
GREEN = *scalars*	Green intensity of each colour (between 0 and 255)
BLUE = *scalars*	Blue intensity of each colour (between 0 and 255)
MATCH = *scalars*	Number of a Genstat colour to define any unset values of RED, GREEN or BLUE; default is to restore the original values of the colour
SAVE = *pointers*	Pointers each containing three scalars to save the red, green and blue intensities of the colours

Description

The COLOUR directive allows you to redefine the standard Genstat colours. In Releases prior to Release 11 these standard colours were used in directives like PEN to define the colours of the various components of the graph. In Release 11, however, colours in these directives are defined by default by setting them explicitly to an RGB value (see PEN). However, you can arrange to use the old method by specifying the statement

```
SET [CMETHOD=standard]
```

Genstat uses the RGB colour system to define each standard colour (numbered from 0 to 256) in terms of its red, green and blue components. These are specified as integer values in the range [0,255]. Thus black is represented by (0,0,0), white by (255,255,255), red by (255,0,0), and so on. For compatibility with earlier releases, fractional values between 0 and 1 will be multiplied by 255. The COLOUR directive can be used in three ways. Firstly you can define a colour in RGB terms. For example, you could put

```
COLOUR 1; RED=255; BLUE=0; GREEN=255
```

to define colour 1 as yellow. Points plotted in colour 1 would then appear as yellow. Alternatively, the MATCH parameter allows a colour to take its RGB values from the current settings of another colour. For example,

```
COLOUR 2; MATCH=1
```

will set colour 2 also to be yellow. Note that if colour 1 is changed again, colour 2 will not be altered. Finally a colour can be returned to its initial default settings by specifying only the colour number. For example,

```
COLOUR 1,2
```

will set colours 1 and 2 back to their original values. The background colour may be altered by changing the definition of colour 0.

The exact effects of the COLOUR directive will vary for different graphics devices. For monochrome devices, the colour (0,0,0) is taken as the background colour, and any other combination is taken as the foreground colour.

By default any parameters that are not mentioned explicitly in the statement are left unchanged, but you can specify option RESET=yes to reset them back to their initial default settings.

Option: RESET.
Parameters: NUMBER, RED, GREEN, BLUE, MATCH, SAVE.

See also
Directive: PEN.
Procedure: GETRGB.
Genstat Reference Manual 1 Summary section on: Graphics.

COMBINE

Combines or omits "slices" of a multi-way data structure (table, matrix or variate).

Options

`OLDSTRUCTURE` = *identifier*	Structure whose values are to be combined; no default i.e. this option must be set
`NEWSTRUCTURE` = *identifier*	Structure to contain the combined values; no default i.e. this option must be set

Parameters

`OLDDIMENSION` = *factors* or *scalars*
: Dimension number or factor indicating a dimension of the `OLDSTRUCTURE`

`NEWDIMENSION` = *factors* or *scalars*
: Dimension number or factor indicating the corresponding dimension of the `NEWSTRUCTURE`; this can be omitted if the dimensions are in numerical order, while zero settings (each in conjunction with a single `OLDPOSITION`) allows a slice of an old table to be mapped into a new table with fewer dimensions

`OLDPOSITIONS` = *pointers*, *texts*, *variates* or *scalars*
: These define positions in each `OLDDIMENSION`: pointers are appropriate for matrices whose rows or columns are indexed by a pointer; texts are for matrices indexed by a text, variates with a textual labels vector, or tables whose `OLDDIMENSION` factor has labels; and variates either refer to levels of table factors or numerical labels of matrices or variates, if these are present, otherwise they give the (ordinal) number of the position. If omitted, the positions are assumed to be in (ordinal) numerical order. Margins of tables are indicated by missing values

`NEWPOSITIONS` = *pointers*, *texts*, *variates* or *scalars*
: These define positions in each `NEWDIMENSION`, specified similarly to `OLDPOSITIONS`; these indicate where the values from the corresponding `OLDDIMENSION` positions are to be entered (or added to any already entered there)

`WEIGHTS` = *variates*
: Define weights by which the values from each `OLDDIMENSION` coordinate are to be multiplied before they are entered in the `NEWDIMENSION`

Description

Sometimes you may wish to reclassify a table to have factors different from those that you used in its declaration. `COMBINE` allows you to omit or to combine levels of the classifying factors. Furthermore, if you want to take just one level of a factor, you can copy the values into a table with one less dimensions.

You specify the original table using the `OLDSTRUCTURE` option, and a table to contain the reclassified values using the `NEWSTRUCTURE` option; if you have not already declared the new table, it will be declared implicitly. You must specify both of these options.

You can modify several of the classifying factors at a time. You list the factors of the original table with the `OLDDIMENSION` parameter, and the equivalent factors of the new table with

NEWDIMENSION. An alternative way of doing this is to give a dimension number, specifying the position of the factor in the classifying set of the table; for the NEWDIMENSION list, this requires that you have already declared the new table. You can even omit the list of dimensions if they would be in ascending numerical order. NEWDIMENSION can also be set to 0 (to imply no corresponding new factor), allowing you to extract a single slice of a table into a table with fewer dimensions.

You use the OLDPOSITIONS and NEWPOSITIONS parameters to specify how this combining is to be done. These parameters specify a pair of vectors for each pair of old and new dimensions, listing positions within the old dimension and the corresponding positions to which they are mapped in the new dimension. The positions can be defined in terms of either the levels or the labels of the factor that classifies the dimension. If you omit the vector for one of the dimensions, it is assumed to contain each value once only, taken in the order in which they occur in the levels vector of the factor. You indicate a margin of the table by a missing value in a variate, or by a null string in a text. Values in the original table can be allocated to more than one place. In parallel with the vectors of positions, you can also use the WEIGHTS parameter to specify a variate of weights by which the values are multiplied before being entered into the new table.

Although the main way in which you will use COMBINE is likely to be for tables, you can also use it on rectangular matrices and even variates. For these, the dimensions can only be numbers: number 1 refers to the rows of a matrix, and 2 to the columns; number 1 refers to the rows (or units) of a variate. The position vectors refer to the labels vectors of matrices, which can be variates, texts or pointers; or they refer to the unit labels of a variate, which can be held in either a variate or a text. If a dimension has no labels vector, you use a variate to specify its positions; then each value of the variate gives the number of a row, column or unit. You can do the same also if the labels vector is something other than a variate: that is, a text or a pointer.

Options: OLDSTRUCTURE, NEWSTRUCTURE.
Parameters: OLDDIMENSION, NEWDIMENSION, OLDPOSITIONS, NEWPOSITIONS, WEIGHTS.

Action with RESTRICT
Any restrictions, for example on OLDPOSITIONS or NEWPOSITIONS variates, are ignored.

See also
Directives: TABLE, TABULATE, MARGIN.
Procedures: MTABULATE, PERCENT, SVSTRATIFIED, SVTABULATE, TABINSERT, TABMODE, TABSORT.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

COMMANDINFORMATION

Provides information about whether (and how) a command has been implemented.

No options

Parameters

NAME = *texts*	Single-line texts supplying the names of the commands
IMPLEMENTATION = *texts*	Single-line texts set to 'directive', 'procedure' or a null string ('') according to the type of command
CHANNEL = *scalars*	Saves the channel for a procedure from a procedure library
PRESENTNOW = *scalars*	Logical set to one if the command is now present, or zero otherwise

Description

COMMANDINFORMATION enables you to discover whether a command is present in your version of Genstat and, if so, whether it is a directive or a procedure.

The name of the command must be supplied in a single-value text, using the NAME parameter. The IMPLEMENTATION parameter can save another single-valued text, which is set to 'directive' or 'procedure' according to the type of command. If the command is not present, it is set to a null string ('') .

The PRESENTNOW parameter provides another, possibly simpler, way of discovering whether the directive or procedure is currently present within Genstat. This saves a scalar containing the value one if the command is present, or zero otherwise.

For procedures accessed from a procedure library, the CHANNEL option can save a scalar with the number of the channel to which the library is attached. This contains a missing value if the command is not present as a procedure. It contains zero if the procedure was created in this job (using the PROCEDURE directive). The channel number for the official procedure library is 12, and the channel for the local procedure library is 11.

Parameters: NAME, IMPLEMENTATION, CHANNEL, PRESENTNOW.

See also

Directives: OPEN, PROCEDURE, SYNTAX.
Procedures: SPSYNTAX.
Genstat Reference Manual 1 Summary section on: Program control.

CONCATENATE

Concatenates and truncates lines (units) of text structures; allows the case of letters to be changed.

Options

NEWTEXT = *text*	Text to hold the concatenated/truncated lines; default is the first OLDTEXT vector
CASE = *string token*	Case to use for letters (given, lower, upper, changed); default give leaves the case of each letter as given in the original string

Parameters

OLDTEXT = *texts*	Texts to be concatenated
WIDTH = *scalars* or *variates*	Number of characters to take from the lines of each text, a negative value takes all the (unskipped) characters other than trailing spaces; if * or omitted, all the (unskipped) characters are taken
SKIP = *scalars* or *variates*	Number of characters to skip at the left-hand side of the lines of each text, a negative value skips all initial spaces; if * or omitted, no characters are skipped

Description

The CONCATENATE directive joins lines of several texts together, side by side, to form a new text. You can specify the identifier of this text by the NEWTEXT option, in which case it need not already have been declared as a text. If you do not specify NEWTEXT, Genstat places the new textual values into the first text in the OLDTEXT parameter list (replacing its existing values).

The texts to be concatenated are specified by OLDTEXT; they should all contain the same number of lines, unless you want to insert an identical series of characters into every line of the new text: a series of characters that is to be duplicated within every line can be specified either as a string, or in a single-valued text.

If you give a variate in the SKIP list, then it must contain a value for each line of the text in the OLDTEXT list; the value indicates the number of characters to be omitted at the beginning of that line. Alternatively, you can give a scalar if the same number of characters is to be omitted at the start of every line. Similarly the WIDTH parameter specifies how many characters are to be taken, after omitting any initial characters as specified by SKIP.

CONCATENATE also provides easy ways of removing spaces at the beginning or the end of strings. A negative value of the SKIP parameter deletes all the spaces at the start of a string, while a negative value of the WIDTH parameter deletes all the spaces at the end of a string.

The CASE option enables you to change the case of letters. By default, CASE=given to leave the case of each letter as given in the existing text. To change all letters to upper case (or capitals) you can put CASE=upper, or CASE=lower to change all letters to lower case. Alternatively, CASE=changed puts lower-case letters into upper case, and upper-case letters into lower case!

Options: NEWTEXT, CASE.
Parameters: OLDTEXT, WIDTH, SKIP.

Action with RESTRICT

CONCATENATE takes account of restrictions on any of the vectors that occur in the statement. If more than one vector is restricted, then each such restriction must be the same. The values of the units that are excluded by the restriction are left unchanged.

See also
Directives: TEXT, EDIT, TXBREAK, TXCONSTRUCT, TXFIND, TXPOSITION, TXREPLACE.
Procedure: APPEND, SUBSET, STACK, UNSTACK.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

CONTOUR

Produces contour maps of two-way arrays of numbers on the terminal or line printer (synonym of LPCONTOUR).

Options

CHANNEL = *scalar*	Channel number of output file; default is current output file
INTERVAL = *scalar*	Contour interval for scaling; default * i.e. determined automatically
TITLE = *text*	General title; default *
YTITLE = *text*	Title for y-axis; default *
XTITLE = *text*	Title for x-axis; default *
YLOWER = *scalar*	Lower bound for y-axis; default 0
YUPPER = *scalar*	Upper bound for y-axis; default 1
XLOWER = *scalar*	Lower bound for x-axis; default 0
XUPPER = *scalar*	Upper bound for x-axis; default 1
YINTEGER = *string token*	Whether y-labels integral (yes, no); default no
XINTEGER = *string token*	Whether x-labels integral (yes, no); default no
LOWERCUTOFF = *scalar*	Lower cut-off for array values; default *
UPPERCUTOFF = *scalar*	Upper cut-off for array values; default *

Parameters

GRID = *identifiers*	Pointers (of variates representing the columns of a data matrix), matrices or two-way tables specifying values on a regular grid
DESCRIPTION = *texts*	Annotation for key

Description

The CONTOUR directive has been replaced by the LPCONTOUR directive, and may be removed in a future release or modified to produce high-resolution plots instead of character-based plots.

A contour plot provides a way of displaying three-dimensional data in a two-dimensional plot. The data values are supplied as a rectangular array of numbers that represent the values of the variable in the third dimension, often referred to as *height* or the *z-axis*. The first two dimensions (x and y) are the rows and columns indexing the array; the complete three-dimensional data set is referred to as a *surface* or *grid*. Contours are lines that are used to join points of equal height, and usually some form of interpolation is used to estimate where these points lie. The resulting contour plot is not necessarily very "realistic" when compared to perspective plots produced by DSURFACE, but it has the advantage that the entire surface can easily be examined, without the danger of some parts being obscured by high points or regions.

You might use contour plots for example when you have data sampled at points on a regular grid, such as the concentrations of a trace element or nutrient in the soil. Contours are also very useful when fitting nonlinear models, when they can be used to study two-dimensional slices of the likelihood surface, to help find good initial estimates of the parameters.

The CONTOUR directive produces output for a line printer by using cubic interpolation between the grid points to estimate a z-value for each character position in the plot. Each value is reduced to a single digit in the range 0 ... 9, according to the rules described below. To produce the contour plot only the even digits are printed: you can then see the contours as the boundaries between the blank areas and the printed digits.

The GRID parameter can be set to a matrix, a two-way table (with the first factor defining the rows), or a pointer to a set of variates each containing a column of data. We explain the conventions in terms of a matrix as input, but similar rules apply to the other structures. When

reading or printing a matrix the origin of the rows and columns (row 1, column 1) appears at the top left-hand corner. However, in forming the contour plot the rows are reversed in order so that the first row of the matrix is placed at the bottom of the contour; thus the origin of the contour is located, according to the usual conventions, at the bottom left-hand corner of the plot. The DCONTOUR directive also reverses the rows of the grid in the same way.

CONTOUR scales the grid values by dividing by the contour interval. The scaled grid values are then converted to single digits by taking the remainder modulo 10 and truncating the fractional part. The INTERVAL option allows you to set the contour interval. For example, if the grid values range from 17 to 72 and the interval is set to 10, contour lines (the boundaries between blank space and printed digits) will occur at grid values of 20, 30, 40, 50, 60 and 70. By default, the interval is determined from the range of the data in order to obtain 10 contours.

The UPPERCUTOFF and LOWERCUTOFF options can be used to define a window for the grid values that will form the contours. All values above or below these are printed as X. Setting either UPPERCUTOFF or LOWERCUTOFF will change the default contour interval, as the range of data values is effectively curtailed.

You can use the TITLE, YTITLE and XTITLE option to annotate the contour plot. If you specify several grids, these will be plotted in separate frames and the text of the TITLE option will appear at the top of each one. You should thus use TITLE only to give a general description of what the contours represent. The DESCRIPTION parameter can be used to add specific descriptions to be printed at the bottom of each individual plot.

The YUPPER and YLOWER options allow you to set upper and lower bounds for the y-axis; thus generating axis labels that reflect the range of values over which the grid was observed or evaluated. Setting YINTEGER=yes will ensure the labels are printed as integers, if possible. The default axis bounds are 0.0 and 1.0. The options XLOWER, XUPPER and XINTEGER similarly control labelling of the x-axis.

Options: CHANNEL, INTERVAL, TITLE, YTITLE, XTITLE, YLOWER, YUPPER, XLOWER, XUPPER, YINTEGER, XINTEGER, LOWERCUTOFF, UPPERCUTOFF.
Parameters: GRID, DESCRIPTION.

Action with RESTRICT

CONTOUR takes account of restrictions on any of the variates in a GRID pointer.

See also

Directives: DCONTOUR, LPCONTOUR.
Genstat Reference Manual 1 Summary section on: Graphics.

COPY

Forms a transcript of a job.

Option

PRINT = *string tokens*	What to transcribe (statements, output); default stat

Parameter

scalar	Channel number of output file

Description

The COPY directive can be used to save a copy of either input statements, or output, or both, in an output file. For example

```
OPEN 'GEN.REC','GEN.OUT'; CHANNEL=2,3; FILETYPE=output
COPY [PRINT=statements] 2
COPY [PRINT=output] 3
```

will keep a record of all the statements in the file GEN.REC and of all the output in the file GEN.OUT.Setting PRINT=* stops any copying to the specified channel. For example

```
COPY [PRINT=*] 2
```

stops copying to GEN.REC.

You can thus obtain output in more than one style (for example RTF and HTML as well as plain-text) by opening, and then copying, to files in the required styles (see OPEN).

Option: PRINT.

Parameter: unnamed.

See also

Directives: OPEN, PRINT, READ.

Genstat Reference Manual 1 Summary section on: Input and output.

CORRELATE

Forms correlations between variates, autocorrelations of variates, and lagged cross-correlations between variates.

Options

PRINT = *string tokens*	What to print (correlations, autocorrelations, partialcorrelations, crosscorrelations); default *
GRAPH = *string tokens*	What to display with graphs (autocorrelations, partialcorrelations, crosscorrelations); default *
MAXLAG = *scalar*	Maximum lag for results; default * i.e. value inferred from variates to save results
CORRELATIONS = *symmetric matrix*	Stores the correlations between the variates specified by the SERIES parameter

Parameters

SERIES = *variates*	Variates from which to form correlations
LAGGEDSERIES = *variates*	Series to be lagged to form crosscorrelations with first series
AUTOCORRELATIONS = *variates*	To save autocorrelations, or to provide them to form partial autocorrelations if SERIES=*
PARTIALCORRELATIONS = *variates*	To save partial autocorrelations
CROSSCORRELATIONS = *variates*	To save crosscorrelations
TESTSTATISTIC = *scalars*	To save test statistics
VARIANCES = *variates*	To save prediction error variances
COEFFICIENTS = *variates* or *matrices*	To save prediction coefficients: in a variate to keep only those for the maximum lag, or in a matrix to keep the coefficients for all lags up to the maximum

Description

The most straightforward use of the CORRELATE directive is to calculate correlation coefficients between a set of variates. For example this would display the correlations between the variates Age, Height and Weight as a lower-triangular matrix.

```
CORRELATE [PRINT=correlations; CORRELATIONS=Corr] \
  Age,Height,Weight
```

The correlations are also saved in the symmetric matrix Corr using the CORRELATIONS option. Note that, if there are missing values, CORRELATE uses only those units where none of the variates is missing.

CORRELATE can also be used to obtain autocorrelations of a time series, that is the correlations between values in the series lagged by particular time intervals. The set of autocorrelations for all possible lags is the *autocorrelation function*. You can derive the *partial autocorrelation function* from these. To look at the relationship between two series, you should use the *cross-correlation function* between one series and the other lagged by the various intervals. The sample autocorrelation function of a series can be displayed either as a table of numbers, or as a graph – called a *correlogram*. In either case, you must specify the maximum lag for which the autocorrelation is to be calculated, *m* say. You can do this either by setting the MAXLAG option to *m*, or by pre-defining the length of a variate to be *m*+1 and including it in the

AUTOCORRELATIONS parameter to store the calculated values. Genstat includes the autocorrelation at lag 0 in the autocorrelation function; this is always unity. The formula used for the sample autocorrelation at lag k is

$$r_k = (1 - k/n) \times C_k / C_0$$

where

$$C_k = (1 / n_k) \sum_{i=1 \ldots n-k} \{(y_t - \text{mean}(y))\,(y_{t+k} - \text{mean}(y))\}$$

The number n_k is the number of terms included in the sum. The series can contain missing values, but the calculation excludes any product that involves any missing values at all. You can restrict a series, but the restricted set must consist of a contiguous set of units. Thus, you can look at the autocorrelation function derived from just the first section of a series, or from just the last section, or from a section in the middle; but you cannot use restriction to exclude a section from the middle of the series, or to exclude just individual observations.

The AUTOCORRELATIONS parameter allows you to save the calculated autocorrelations. If you want to display a correlogram in a different form from the standard one produced by the GRAPH option, you must save the autocorrelations and plot them explicitly using either the GRAPH or DGRAPH directives. You will then need to define the variate of lags from *0* to *m*.

The TESTSTATISTIC parameter of CORRELATE allows you to save a statistic that can be used to test the hypothesis that the true autocorrelation is zero for positive lags. It is defined as

$$S = n \sum_{k=1 \ldots m} \{ r_k^2 \}$$

Provided n (the number of data values) is large and m (the maximum lag) is much smaller than n, then under the null hypothesis, the statistic has a chi-square distribution with m degrees of freedom. Thus, a large value provides evidence of autocorrelation in a time series.

You can calculate autocorrelation functions for several series in one statement by specifying several variates with the SERIES parameter.

Genstat forms partial autocorrelations from an autocorrelation function. The value at lag k is defined as

$$\text{corr}(y_t, y_{t-k} \mid y_{t-1}, y_{t-2} \ldots y_{t-k+1})$$

representing the excess correlation between values separated by k timepoints that is not accounted for by the intermediate points; it is denoted by $\varphi_{k,k}$ because it is also the value of the last in the set of coefficients in the autoregressive prediction equation:

$$y_t = c + \varphi_{k,1} y_{t-1} + \ldots + \varphi_{k,k} y_{t-k} + e_{k,t}$$

Genstat calculates these coefficients recursively for k=1...m by

$$\varphi_{k,k} = (r_k - \varphi_{k-1,1} r_{k-1} - \ldots - \varphi_{k-1,k-1} r_1) / v_{k-1}$$

$$\varphi_{k,j} = \varphi_{k-1,j} - \varphi_{k,k} \varphi_{k-1,k-j}\,,\ j=1 \ldots k-1$$

$$v_k = v_{k-1} (1 - \varphi_{k,k}^2)$$

It starts with v_0=1, the quantity v_k being the kth order prediction error variance ratio

$$\text{variance}(e_{k,t}) / \text{variance}(y_t).$$

Partial correlations provide a valuable alternative way of displaying the autocorrelation structure of a series. You can display the partial autocorrelation function either as a table of numbers, or as a graph. Two methods are available for doing this. You can supply the series using the SERIES parameter, in which case the autocorrelations are formed first, automatically, and the partial autocorrelations are then derived from them. Alternatively, you can set SERIES=*, and provide the autocorrelations using the AUTOCORRELATIONS parameter. You can specify the maximum lag, either by setting the MAXLAG option, or by pre-defining the length of a variate specified for either the AUTOCORRELATIONS or the PARTIALCORRELATIONS parameter.

You can save the partial autocorrelation function using the PARTIALCORRELATIONS parameter. You can set the VARIANCES and COEFFICIENTS parameters to variates to save the *prediction-error variances* $v_0 \ldots v_m$, and the *prediction coefficients 1,* $\varphi_{m,1} \ldots \varphi_{m,m}$ for the maximum lag m. Genstat sets the first coefficient to 1, and also the first element of the partial autocorrelation sequence to 1: you should find this to be a useful convention for the lag 0 values. Alternatively, if the COEFFICIENTS parameter is set to a matrix structure, the rows of this

matrix will be used to save the prediction coefficients for *all* the orders up to the maximum lag.

CORRELATE will print a warning if you include missing values in an autocorrelation function that you have supplied, or if for some other reason the autocorrelations are invalid. In particular, if a partial autocorrelation value is obtained outside the range (−1, 1), Genstat will truncate the sequence at the previous lag.

You can calculate cross-correlations between two series by specifying one series with the SERIES parameter and the other with the LAGGEDSERIES parameter. You must define the maximum lag, as for autocorrelations, and you can again plot or tabulate the resulting function. Missing values are allowed, as for autocorrelations. Genstat calculates the sample cross-correlation between the first series x_t and the lagged series y_t at lag k using:

$$r_k = (1 - k/n)\, C_k / (s_x\, s_y)$$

where

$$C_k = (1 / n_k) \sum_{i=1 \ldots n-k} \{(x_t - \text{mean}(x))\,(y_{t+k} - \text{mean}(y))\}$$

The series x_t and y_t may be of different lengths. The summation includes all possible terms, but excludes any product containing missing values; the number n_k is the number of terms included in the sum. The values $\bar{x}$ and $\bar{y}$ are the sample means, and s_x, s_y are the sample standard deviations. The number n is the minimum of the number of values of x and of y, excluding missing values. You can restrict either series to a set of contiguous units: if both are restricted, their restrictions must match.

You can save the cross-correlation function using the CROSSCORRELATIONS parameter. You can also save a test statistic using the TESTSTATISTIC parameter; this is used similarly to the statistic to test for lack of lagged cross-correlation in one direction of the relationship between two series. However the test is valid only if each of the series has a zero autocorrelation function. Cross-correlations take precedence in the storage. Thus if you request both autocorrelations and cross-correlations in a single CORRELATE statement, the stored test statistic will relate to the cross-correlations: that for the autocorrelations will not be stored.

Options: PRINT, GRAPH, MAXLAG, CORRELATIONS.

Parameters: SERIES, LAGGEDSERIES, AUTOCORRELATIONS, PARTIALCORRELATIONS, CROSSCORRELATIONS, TESTSTATISTIC, VARIANCES, COEFFICIENTS.

Action with RESTRICT

You can restrict the units involved in the calculation of the correlations by restricting either the SERIES variate, or the LAGGEDSERIES variate (if present). For the calculation of autocorrelations, partial-correlations or cross-correlations, the restriction must define a contiguous set of units. If SERIES and LAGGEDSERIES are both restricted, they must be restricted in exactly the same way.

See also

Directives: FSSPM, TSM.

Procedures: DCORRELATION, FCORRELATION, PRCORRELATION, PARTIALCORRELATIONS, FVCOVARIANCE.

Functions: CORRELATION, COVARIANCE. VARIANCE.

Genstat Reference Manual 1 Summary sections on: Basic and nonparametric statistics, Calculations and manipulation, Time series.

COUNTER

Increments a multi-digit counter using non base-10 arithmetic.

Options

NREQUIRED = *scalar*	Specifies the number of values required for the counter; default 2
NFOUND = *scalar*	Saves the number of counter values that could be formed
DIRECTION = *string token*	Specifies the direction of the sequence of increments to the counter (ascending, descending); default asce

Parameters

START = *scalars*	Provides the starting values for the digits in the counter
END = *scalars*	Can provide values to define the end of the sequence of counter values
STEP = *scalars*	Specifies the amount by which to increment each digit of the counter
BASE = *scalars*	Specifies the base of the numbers used for each digit
DIGITSEQUENCE = *variates*	Saves the sequence of values generated for each digit

Description

COUNTER is useful if you want to increment a counter made up of several digits that recycle to limits that may be different from ten. For example, times in seconds, minutes and hours, or measurements in inches, feet and yards.

The parameters provide details of the digits in the counter, all in scalars. The BASE parameter specifies the base of the numbers used for each digit (e.g. 60 for seconds and minutes, and 24 for hours). The START parameter supplies the starting values of the digits, ranging from zero to BASE minus one. The STEP parameter specifies the size of the increment for each digit. The digits are updated from the right-hand side and, when one goes beyond its limit, the next one is incremented by an extra value of one for an ascending sequence, or minus one for a descending sequence. The DIGITSEQUENCE saves the sequence of values formed for each digit of the counter, in variates.

The END parameter can specify values to define the end of the sequence. If a value is specified for every digit, the sequence ends when the next set of digits would go beyond those supplied by END: above END for an ascending sequence, or below for a descending sequence. (See the DIRECTION option.) Otherwise, the sequence ends when all the digits would go beyond their limits: BASE minus one for an ascending sequence, or zero for a descending sequence.

The NREQUIRED option specifies the number of values that are required for the counter. The default is 2, i.e. START and one other. The NFOUND option can save the number of values that have been formed. The DIRECTION option controls whether the sequence of counter values should be regarded as ascending or descending, when checking for the end of the sequence. The default is ascending.

Options: NREQUIRED, NFOUND, DIRECTION.
Parameters: START, END, STEP, BASE. DIGITSEQUENCE.

See also

Genstat Reference Manual 1 Summary section on: Program control.

COVARIATE

Specifies covariates for use in subsequent ANOVA statements.

No options

Parameter

variates or *pointers* Covariates

Description

To perform analysis of covariance you need to define the treatment model (using TREATMENTSTRUCTURE) and the underlying structure of the design (using BLOCKSTRUCTURE) as in ordinary analysis of variance, and then simply specify the required covariates using the COVARIATE directive. You can then do the analysis by ANOVA, get further output by ADISPLAY and so on, in the usual way.

In the simplest form of the COVARIATE directive, its (unnamed) parameter just contains a list of the variates that are to be used as covariates. Alternatively, you can group some of the variates into pointers. The analysis-of-variance table will then contain a line for each group instead of the individual covariates in that group (see below).

You can use covariates to incorporate any quantitative information about the units into the model. In field experiments there may often be linear trends in fertility. These can be estimated and removed by fitting a covariate of the position of the plot along the direction of the trend. For example

```
COVARIATE Location
```

For a quadratic trend, you would also include a covariate containing the squares of the positions.

```
CALCULATE Quadtrend = Location**2
COVARIATE Location,Quadtrend
```

In experiments on animals, you may wish to use measurements such as the original weight. However the assumption is always that the y-variate is linearly related to the covariates.

Covariates are incorporated into the model as terms for a linear regression. Genstat fits the covariates, together with the treatments, in each stratum. This should explain some of the variability of the units in the stratum, and so decrease the stratum residual mean square.

Each treatment combination will have been applied to units whose mean value for each covariate differs from that of other treatment combinations; so even in the absence of any treatment effects, the y-values recorded for the different combinations would not be identical. A further effect of the analysis is to adjust the treatment estimates for the covariates, to correct for this. This adjustment causes some loss of efficiency in the treatment estimation. The remaining efficiency is measured by the *covariance efficiency factor*, shown for each treatment term in the "cov. ef." column of the analysis-of-variance table. The values are in the range zero to one. A value of zero indicates that the treatment contrasts are completely correlated with the covariates: after the covariates have been fitted there is no information left about the treatments. A value of one indicates that the covariates and the treatment term are orthogonal. Usually the values will be around 0.8 to 0.9. A low value should be taken as a warning: either the measurements used as covariates have been affected by the treatments, which can occur when the measurements on covariates are taken after instead of before the experiment, or the random allocation of treatments has been unfortunate in that some treatments are on units with generally low values of the covariates while others are on generally high ones. The covariance efficiency factor is analogous to the efficiency factor printed for non-orthogonal treatment terms; details of its derivation can be found in Payne & Tobias (1992).

For a residual line in the analysis of variance, the value in the "cov. ef." column measures how much the covariates have improved the precision of the experiment. This is calculated by dividing the residual mean square in the unadjusted analysis (which excludes the covariates) by

its value in the adjusted analysis.

The covariance efficiency factor is used by Genstat in the calculation of standard errors for tables of effects; if you want to calculate the net effect of the analysis of covariance on the precision of the estimated effects of a treatment term, you should multiply the covariance efficiency factor of the term by the value printed in the residual line of the stratum where the term is estimated. Where a term has more than one degree of freedom, the adjustment given by the covariance efficiency factor is an average over all the comparisons between the effects of the term. However this adjustment should not differ by much from those required for any particular comparison unless the randomization has been especially unfortunate. For a table of means classified by several factors, Genstat combines the covariance efficiency factors of the effects from which the means are calculated into a harmonic mean, weighted according to the numbers of degrees of freedom of each term.

The adjusted analysis-of-variance table has an extra line in each stratum, giving the sum of squares due to the covariates. This is the extra sum of squares that is removed by the covariates after eliminating all that can be explained by the treatments. It thus lets you assess whether there is any evidence that the covariates are required in the model. If there are several covariates Genstat will also print their individual contributions to that sum of squares, giving first the sum of squares that can be explained by the first covariate in the `COVARIATE` list, then the extra sum of squares that can be accounted for by fitting the second covariate, and so on. However, if some of the covariates were grouped together into a pointer in the `COVARIATE` list, their contributions will be pooled into a single line.

The line for each treatment term in the analysis-of-variance table contains the sum of squares eliminating the covariates. It indicates whether there is evidence of any effects of that term, after taking account of the differences in the values of the covariates on the units to which each treatment was applied.

The method that Genstat uses for analysis of covariance essentially reproduces the method that you would use if you were doing the calculations by hand. First of all, it analyses each covariate according to the block and treatment models. You can print information from these analyses using the `CPRINT` option of either `ANOVA` or `ADISPLAY`. As `ADISPLAY` does not constrain you to list save structures that were all produced by the same `ANOVA`, `CPRINT` will produce information about the covariate analyses from every save structure that you list; duplicate information will thus be produced if several of the save structures are for analyses involving the same covariates. The output from `CPRINT`, particularly the analysis-of-variance table, gives you another way of assessing the relationship between treatments and covariates: a large variance ratio for a treatment term in the analysis of one of the covariates would indicate either that the treatment had affected the covariate or that the randomization had been unfortunate (as discussed in the description of cov. ef. above).

Genstat then analyses each y-variate in turn. First of all it does the usual analysis ignoring the covariates. You can control output from this unadjusted analysis by the `UPRINT` option of `ANOVA` and `ADISPLAY`. (So the whole of the output given for the example could have been produced by a single `ANOVA` statement.) Then the covariates are fitted by linear regression and the full, adjusted, analysis is calculated. Output from the adjusted analysis is controlled by the `PRINT` option of `ANOVA` and `ADISPLAY`. This option has an extra setting, not available for `UPRINT` and `CPRINT`: `PRINT=covariates` prints the regression coefficients of the covariates as estimated in each stratum.

Options: none.
Parameter: unnamed.

Reference

Payne, R.W. & Tobias, R.D. (1992). General balance, combination of information and the analysis of covariance. *Scandinavian Journal of Statistics*, **19**, 3-23.

See also

Directives: ANOVA, BLOCKSTRUCTURE, TREATMENTSTRUCTURE, ADISPLAY, AKEEP.
Procedures: AFCOVARIATES, ASTATUS, AUNBALANCED.
Genstat Reference Manual 1 Summary section on: Analysis of variance.

CVA

Performs canonical variates analysis.

Options

PRINT = *string tokens*	Printed output required (roots, loadings, means, residuals, distances, tests); default * i.e. no printing
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no

Parameters

WSSPM = *SSPMs*	Within-group sums of squares and products, means etc (input for the analyses)
LRV = *LRVs*	Saves loadings, roots and trace from each analysis
SCORES = *matrices*	Saves canonical variate means
RESIDUALS = *matrices*	Saves distances of the means from the dimensions fitted in each analysis
DISTANCES = *symmetric matrices*	Saves inter-group-mean Mahalanobis distances
ADJUSTMENTS = *matrices*	Saves the adjustment terms
SAVE = *pointers*	Saves details of the analysis; if unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive)

Description

You specify the input for CVA using its first parameter, WSSPM, this may contain a list of structures, in which case Genstat repeats the analysis for each of them. The input must be an SSPM structure, declared with the GROUPS option of the SSPM directive set to a factor giving the grouping of the units. If the variates used to form this SSPM structure are restricted, then the SSPM is restricted in the same way, and so the CVA directive takes account of the restriction. The SSPM contains information on the within-group sums of squares and products, pooled over all the groups; it also contains the group means and group sizes, from which Genstat can derive the between-group sums of squares and products. CVA finds linear combinations of the original variables that maximize the ratio of between-group to within-group variation, thereby giving functions of the original variables that can be used to discriminate between the groups. The squares of the printed distances between group means are Mahalanobis D^2 statistics when all the dimensions are used; otherwise they are approximations. You can form exact Mahalanobis distances with the PCO directive.

The three options of the CVA directive control the printed output. By default there is no printed output, and so you should set the PRINT option to indicate which sections you want. Results can be printed for a subset of the latent roots by setting the NROOTS and SMALLEST options of CVA. NROOTS specifies the number of roots for which you want the results to be printed. By default these will be the largest roots, unless you set SMALLEST=yes; then the results will be printed for the smallest non-zero roots. When you print a subset of the results, residuals can be formed and printed from the dimensions that are not displayed.

The significance tests that are printed are for a significant dimensionality greater than k, that is for the joint significance of the first, second, ..., (k+1)th latent roots. This test is printed for k=0, 1, ... min(g−1, v)−1. If the test is "not significant" for k=r, then the values of chi-square for k>r should be ignored as the indication is that the remaining dimensions have no interesting structure. The test statistic (Bartlett 1938) is asymptotically distributed as chi-square with

$(v-k)\times(g-k-1)$ degrees of freedom. Here n is the number of units, g is the number of groups, v is the number of variables, and l_i is the ith latent root. If the coefficient $[n-g-\frac{1}{2}(v-g)]$ is less than zero, there are too few units for the statistics to be calculated and a message is printed to this effect. In any case, the tests should be treated with caution unless $n-g$ is very much larger than v.

The latent vectors, or loadings, are scaled in such a way that the average within-group variability in each canonical variate dimension is 1: thus the within-group variation is equally represented in each dimension. Since the latent roots are the successive maxima of the ratio of between-group to within-group variation, loadings corresponding to roots less than 1 are for dimensions in the canonical variate space that exhibit more within-group variation than between-group variation.

The scores for the means are arranged so that their centroid, weighted by group size, is at the origin. This is done by subtracting a constant term, for each canonical variate dimension, from the scores initially formed as a linear combination of the group means of the original variables. These adjustments can be saved, in a matrix of size one by number of groups, using the ADJUSTMENTS parameter.

If you ask for distances, they are formed from the group mean scores for the canonical variate dimensions that are printed. If results are printed for the full dimensionality, the distances will be Mahalanobis distances between the groups.

The LRV parameter allows you to save the loadings, latent roots and their sum (the trace) in an LRV structure, while the SCORES parameter saves the canonical variate means. If you have declared the LRV already, its number of rows must be the same as the number of variates involved in forming the input SSPM. The number of rows of the SCORES matrix, if previously declared, must be equal to the number of groups.

The number of columns of the LRV and of the SCORES matrix corresponds to the number of dimensions to be saved from the analysis, and this must be the same for both of them. If the structures have been declared already, Genstat will take the larger of the numbers of columns declared for either, and declare (or redeclare) the other one to match. If neither has been declared and option SMALLEST retains the default setting no, Genstat takes the number of columns from the setting of the NROOTS option. Otherwise, Genstat saves results for the full set of dimensions. The trace saved as the third component of the LRV structure, however, will contain the sums of all the latent roots, whether or not they have all been saved. Procedure LRVSCREE can be used to produce a "scree" diagram which can be helpful in deciding how many dimensions to save.

The RESIDUALS parameter allows you to save the distances of the means from the dimensions fitted in the analysis in a matrix with number of rows equal to the number of groups and one column. If the latent roots and vectors (loadings) are saved from the analysis, the residuals will correspond to the dimensions not saved; the same applies if you save scores. If neither the LRV nor scores are saved, the saved residuals will correspond to the smallest latent roots not printed.

The DISTANCES parameter allows you to save the inter-group-mean Mahalanobis distances in a symmetric matrix.

The SAVE parameter can supply a pointer to save a multivariate save structure contining all the details of the analysis. If this is unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive). Alternatively, you can set SAVE=* to prevent any save structure being formed if, for example, you have a very large data set and want to avoid committing the storage space.

Options: PRINT, NROOTS, SMALLEST.
Parameters: WSSPM, LRV, SCORES, RESIDUALS, DISTANCES, ADJUSTMENTS, SAVE.

Reference

Bartlett, M.S. (1938). Further aspects of the theory of multiple regression. *Proceedings of the Cambridge Philosophical Society*, **34**, 33-40.

See also

Directives: FCA, MDS, PCO, PCP, ROTATE, FSSPM, SSPM.

Procedures: CVASCORES, CVAPLOT, LRVSCREE, DBIPLOT, DMST, DISCRIMINATE, MANOVA, QDISCRIMINATE, RMULTIVARIATE, SDISCRIMINATE.

Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

DBITMAP

Plots a bit map of RGB colours.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the graph; default 1
YORIENTATION = *string token*	Y-axis orientation of the plot (reverse, normal); default reve
GRIDMETHOD = *string token*	How to draw a grid around the elements of the matrix (present, complete); default * i.e. none
PENGRID = scalar	Pen to use for the grid; default −7
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

BITMAP = *symmetric matrix*, *matrix*, *table*, *pointer to variates* or *variate*	Data to be plotted
ROWS = *variate*	Row indexes for a BITMAP variate
COLUMNS = *variate*	Column indexes for a BITMAP variate

Description

DBITMAP plots a 2-dimensional bit map of RGB colours. These use a standard way of representing a colour, as a single number whose bits are partitioned into three ranges to record the three component colours of red, blue and green (see the PEN directive for more details). They can be read in to Genstat from devices such as scanners, or calculated using the RGB function.

The data are specified by the BITMAP parameter. Data values in a regular two-way grid can be specified by supplying their RGB colours in either a matrix, a symmetric matrix, a 2-way table or a pointer to a set of variates. Alternatively, you can specify irregular data by setting BITMAP to a variate of colours, and the ROWS and COLUMNS parameters to variates defining their row and column indexes.

The GRIDMETHOD option allows you to draw an outline around each element of the plot. The present setting produces an outline for all values that are present; i.e. it ignores missing values. This is suitable where data have been sampled over an irregularly shaped area. Alternatively, with the complete setting, an outline is drawn around every element. By default, no grid is drawn. The PENGRID option specifies which pen to use to draw the grid. The default is to use pen −7.

The YORIENTATION option controls the orientation of the y-axis. By default this is reversed, so that the data are in the same order as they would take if the data matrix were printed.

The TITLE, WINDOW, SCREEN and ENDACTION options are used to specify a title, the plotting window, whether the screen should be cleared first, and whether there should be a pause once the plotting is finished; as in other graphics directives (see, for example, DGRAPH).

Options: TITLE, WINDOW, YORIENTATION, GRIDMETHOD, PENGRID, SCREEN, ENDACTION.
Parameters: BITMAP, ROWS, COLUMNS.

Action with RESTRICT

DBITMAP takes account of restrictions on any of the variates in a BITMAP pointer.

See also

Directives: DCONTOUR, DSHADE, DSURFACE, D3HISTOGRAM, FRAME, XAXIS, YAXIS, PEN, MATRIX, POINTER, SYMMETRICMATRIX, TABLE.

Functions: BLUE, GREEN, GRAY, GREY, RED, RGB.

Genstat Reference Manual 1 Summary section on: Graphics.

DCLEAR

Clears a graphics screen.

Options

DEVICE = *scalar*	Device whose screen is to be cleared; default is to clear the screen of the current graphics device
ENDACTION = *string token*	Action to be taken after clearing the screen (`continue`, `pause`); default `*` uses the setting from the last DEVICE statement

No parameters

Description

DCLEAR clears the screen of a graphics device so that the next plot produced on this device by any of the high-resolution graphics or procedures will be drawn onto an empty screen. All information about the current display, for example axis mappings, is also cleared from memory. The DEVICE option indicates the device to be cleared; by default this is the current graphics device (as set by the DEVICE directive). The ENDACTION option controls what happens after clearing the screen. The default action is the setting specified by the most recent DEVICE statement.

Options: DEVICE, ENDACTION.
Parameters: none.

See also

Directives: DEVICE, DSTART, DFINISH, DKEEP, DDISPLAY.
Genstat Reference Manual 1 Summary section on: Graphics.

DCONTOUR

Draws contour plots on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
YORIENTATION = *string token*	Y-axis orientation of the plot (reverse, normal); default reve
ANNOTATION = *string token*	How to annotate the contours (levels, ordinals); default ordi if there is a key, and leve if there is no key
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *identifier*	Pointer (of variates representing the columns of a data matrix), matrix or two-way table specifying values on a regular grid
PENCONTOUR = *scalar*	Pen number to be used for the contours; default 1
PENFILL = *scalar* or *variate*	Pen number(s) defining how to fill the areas between contours, or 0 to leave the areas in the background colour; default 3
PENHIGHLIGHT = *scalar*	Pen number to use for highlighted contours; default 0 i.e. no highlighting
HIGHLIGHTFREQUENCY = *scalar*	Frequency at which contours are to be highlighted; default 10
NCONTOURS = *scalar*	Number of contours; default 10
CONTOURS = *variate*	Positions of contours
INTERVAL = *scalar*	Interval between contours
DESCRIPTION = *text*	Annotation for key

Description

The contours to be plotted are defined by a grid of z-values or heights. The grid can be a rectangular matrix, a two-way table or a pointer to a set of variates; the y-dimension is represented by the rows of the structure and the x-dimension by the columns. In each case there must be at least three rows and three columns of data (after allowing for any restrictions on a set of variates). Missing values are not permitted; that is, only complete grids can be displayed. If the grid is supplied as a table with margins, these will be ignored when plotting the surface. The YORIENTATION option controls the orientation of the y-axis. By default this is reversed, so that the grid is in the same order as it would be if it were printed.

The WINDOW option defines the window where the contours are plotted, and the KEYWINDOW option similarly specifies where the key should appear. The grid axes are scaled so that the y- and x-dimensions (rows and columns respectively) will match the dimensions of the specified window: if you wish to preserve the "shape" of the grid you should use the FRAME directive to define a window whose y- and x-dimensions are in the same proportions as the grid dimensions.

Titles can be added to these windows using the TITLE and KEYDESCRIPTION options. The SCREEN option controls whether the graphical display is cleared before the histogram is plotted and the ENDACTION option controls whether Genstat pauses at the end of the plot.

The heights of the contour lines are determined using the NCONTOURS, CONTOURS or INTERVAL parameters. The first possibility is to define the contours explicitly using the CONTOURS parameter. Alternatively, if CONTOURS is unset, INTERVAL can set the required interval between each contour. Or, if both CONTOURS and INTERVAL are unset, NCONTOURS defines the required number of lines. Genstat then partitions the range of data values accordingly to give NCONTOURS evenly-spaced contours (or fewer contours if there are insufficient distinct grid values).

The ANNOTATION option controls how the contours are labelled. The default is to label them by integers (ordinals) if there is a key, and by the actual heights (levels) if there is no key. Contour lines that are very short will not be labelled but their height can be determined from adjacent contours. Each line of the key occupies a space of height 0.02 (in normalized device coordinates), and the key window by default has room for a heading and nine contour levels. If necessary, the size of the window can be redefined using the FRAME directive.

The way in which the contour lines are drawn is determined by the pen that has been defined by the PENCONTOUR parameter of DCONTOUR; the default is to use pen 1. The relevant aspects of the pen should be set in advance, if required, using the METHOD, COLOUR, LINESTYLE and THICKNESS parameters of the PEN directive.

If the PENCONTOUR parameter is not used, the plotting method will be line, so that individual contours are made up of straight line segments. If curves are required, METHOD should be set to monotonic to use the method of Butland (1980), or open (or closed) to use the method of McConalogue (1970). Both these methods produce curves that are fitted to independent sets of interpolated points and can thus produce contour lines that cross, particularly if the supplied grid of data is coarse or in a region where the contour height is changing rapidly. If METHOD is set to other values, straight lines will be used to draw the contours.

The PENHIGHLIGHT parameter can specify a pen to use to highlight particular contours. The frequency of the highlighting is then determined by the HIGHLIGHTFREQUENCY parameter; by default every tenth contour is highlighted.

The PENFILL parameter defines how to shade the areas between the contours. If PENFILL is set to zero, there is no shading i.e. the areas between the contours are left in the background colour. If PENFILL is set to a scalar, the shades are defined in increasing intensities of the colour of the specified pen. Alternatively, if PENFILL is set to a variate of length two, the pens are taken to define the shades at the minimum and maximum heights, and the other shades are interpolated between them. Finally, if PENFILL is set to a variate with more than two values, the shading uses the pens in the order in which they are given in the variate (recycling if insufficient pens are defined for the total number of contours). By default, PENFILL=3.

By default, the axis bounds are determined from the grid. Normally the lower bound for each axis will be 1.0 and the upper bound will be the number of rows of the grid for the y-axis, and the number of columns for the x-axis. If a matrix is used to specify the grid, its row and column labels can be set to variates whose values will then be used to determine the axis bounds. The XAXIS and YAXIS directives can be used to control how the axes are drawn or, by setting ACTION=hide, to suppress them altogether.

Options: TITLE, WINDOW, KEYWINDOW, YORIENTATION, ANNOTATION, SCREEN, KEYDESCRIPTION, ENDACTION.

Parameters: GRID, PENCONTOUR, PENFILL, PENHIGHLIGHT, HIGHLIGHTFREQUENCY, NCONTOURS, CONTOURS, INTERVAL, DESCRIPTION.

Action with RESTRICT

DCONTOUR takes account of restrictions on any of the variates in a GRID pointer.

References

Butland, J. (1980). A method of interpolating reasonably-shaped curves through any data. *Proceedings of Computer Graphics*, **80**, 409-422.

McConalogue, D.J. (1970). A quasi-intrinsic scheme for passing a smooth curve through a discrete set of points. *Computer Journal*, **13**, 392-396.

See also

Directives: DBITMAP, DSHADE, DSURFACE, D3HISTOGRAM, FRAME, XAXIS, YAXIS, PEN, MATRIX, POINTER, TABLE.

Procedure: DXYDENSITY.

Genstat Reference Manual 1 Summary section on: Graphics.

DDISPLAY

Redraws the current graphical display.

Options

DEVICE = *scalar*	Device on which to redraw the display (on some systems it may only be possible to redisplay the picture on an interactive graphics device); default uses the current graphics device
ENDACTION = *string token*	Action to be taken after completing the plot (`continue`, `pause`); default `*` uses the setting from the last DEVICE statement

No parameters

Description

This directive is provided to allow additional control of some interactive devices. In some of these, such as PC's operating in full-screen DOS mode, the screen can operate in either text mode or graphics mode. Genstat will automatically switch the screen into the appropriate mode when starting or finishing a graph. Having returned to text mode after examining a graph you may later wish to have another look at the graph that was plotted. DDISPLAY will switch the screen back to graphics mode, thus re-displaying the graph. The ENDACTION option controls what happens after re-displaying the graph; normally with this type of device you would want to pause. The default action for DDISPLAY is the setting specified by the most recent DEVICE statement.

DDISPLAY has no effect when output is directed to a graphics file. For devices that do not operate in this dual-mode fashion, for example a graphics window under X-windows, DDISPLAY has no effect on the graphical display itself. It will however generate a pause if ENDACTION is set to request one.

Note that DDISPLAY does not actually re-plot the graphical output; it merely switches the screen into graphics mode, and assumes that your system has preserved the graphics image.

Options: DEVICE, ENDACTION.
Parameters: none.

See also

Directives: DCLEAR, DEVICE, DKEEP.
Genstat Reference Manual 1 Summary section on: Graphics.

DEBUG

Puts an implicit BREAK statement after the current statement and after every NSTATEMENTS subsequent statements, until an ENDDEBUG is reached.

Options

CHANNEL = *scalar*	Channel number; default 1
NSTATEMENTS = *scalar*	Number of statements between breaks; default 1
FAULT = *string token*	Whether to invoke DEBUG only at the next fault (yes, no); default no

No parameters

Description

The straightforward use of DEBUG causes an immediate break, and then further breaks at regular intervals until you issue an ENDDEBUG statement. Alternatively, by setting option FAULT=yes, you can arrange for Genstat to continue until the next fault diagnostic, and then break. The interval before each further break is specified by the NSTATEMENTS option; by default, breaks take place after every statement.

During the breaks, Genstat takes statements from the channel specified by the CHANNEL option; by default they are taken from channel 1. Each individual break is terminated by an ENDBREAK, exactly like a break invoked explicitly by the BREAK directive.

Options: CHANNEL, NSTATEMENTS, FAULT.
Parameters: none.

See also

Directives: ENDDEBUG, BREAK, CALCULATE.
Genstat Reference Manual 1 Summary section on: Program control.

DECLARE

Declares one or more customized data structures.

Options

`TYPE` = *text*	Single-valued text defining the type of structure to declare
`MODIFY` = *string token*	Whether to modify (instead of redefining) existing structures (`yes`, `no`); default `no`

Parameters

`IDENTIFIER` = *identifiers*	Identifiers of the structures
`VALUES` = *pointers*	Values for each structure
`EXTRA` = *texts*	Extra text associated with each identifier

Description

`DECLARE` is used to set up compound data structures of a customized type. The form of each customized type is defined using the `STRUCTURE` directive. So, for example, after defining a `complex_number` type, by

```
STRUCTURE [NAME='complex_number'] 'real','imaginary'; \
  TYPE='scalar'
```

we can declare `a complex number C` by

```
DECLARE [TYPE='complex_number'] C; VALUES=!p(3,2)
```

The `VALUES` parameter allows values to be defined for the structure, similarly to the `VALUES` parameter of the `POINTER` directive. So, here, the real part of the number `C['real']` is given the value 3, and the imaginary part `C['imaginary']` has the value 2. The `EXTRA` parameter is also used as in the `POINTER` directive, allowing extra text to be associated with the structure for annotation, and the `MODIFY` option allows an existing structure to be modified.

Options: `TYPE`, `MODIFY`.
Parameters: `IDENTIFIER`, `VALUES`, `EXTRA`.

See also

Directives: `STRUCTURE`, `POINTER`, `LRV`, `SSPM`, `TSM`.
Genstat Reference Manual 1 Summary section on: Data structures.

DELETE

Deletes the attributes and values of structures.

Options

REDEFINE = *string token*	Whether or not to delete the attributes of the structures so that the type etc can be redefined (yes, no); default no
LIST = *string token*	How to interpret the list of structures (inclusive, exclusive, all); default incl
PROCEDURE = *string token*	Whether the list of identifiers is of procedures instead of data structures (yes, no); default no
NSUBSTITUTE = *scalar*	Number of times *n* to substitute a dummy in order to determine which structure to delete; default * i.e. full substitution
REMOVE = *string token*	Whether or not to remove the structures from Genstat completely i.e. to delete their identifiers as well as their attributes and values (yes, no); default no

Parameter

identifiers	Structures whose values (and attributes, if requested) are to be deleted

Description

The DELETE directive allows values and attributes of data structures to be deleted so that Genstat can recover the space that they occupy. This may also make the program execute more efficiently as Genstat will then need to keep track of less information. By default only the values are deleted but, if the REDEFINE option is set to yes, the attributes of the structures are also deleted. The only information that is still stored is then the identifier and the internal reference number of the structure. Alternatively, you can set option REMOVE=yes to delete the identifier and reference number as well as the attributes and values, so that no trace of the structure remains.

You may want to delete the attributes merely to save further space. However, the main advantage is that the structures can then be redefined to be of different types.

For example, suppose we have defined a variate Dose by

```
VARIATE [VALUES=0,0,2,2,4,4] IDENTIFIER=Dose
```

This gives Dose the values 0, 0, 2, 2, 4 and 4. If we then put

```
DELETE Dose
```

only the values of Dose are deleted; so we could now assign a new set: for example

```
READ Dose
2 4 0 4 2 0 :
```

Dose remains a variate but now has the values 2, 4, 0, 4, 2 and 0.

Alternatively, if we set REDEFINE=yes in the above example, we could then redefine Dose as (for example) a text with seven values.

```
DELETE [REDEFINE=yes] Dose
TEXT [VALUES=none,double,standard,double,\
  none,standard,none] Dose
```

Once you have defined the type of a structure in a job (as variate, factor or whatever), you cannot redeclare it as a structure of any other type unless you have first used DELETE to delete its values and attributes. The only exception to this rule is that the GROUPS directive also has a REDEFINE option, which allows a variate or text to be redefined as a factor.

The LIST option defines how the parameter list is to be interpreted. With the default setting,

LIST=inclusive, attributes or values are deleted only for the structures in the list. LIST=exclusive means that the parameter list is the complement of the set of structures that are deleted: that is, all named structures that are not in the list are deleted. LIST=all causes the attributes or values of all structures to be deleted. Thus, if LIST=all, any parameter list is ignored; and LIST=exclusive with no parameter is equivalent to LIST=all.

The NSUBSTITUTE option is relevant when the list of structures to delete contains dummies. The default setting, missing value, requests all dummies to be replaced by the structures to which they point (so that those are the structures that are deleted). NSUBSTITUTE allows you to delete dummies instead. If you set NSUBSTITUTE=0, no dummies are substituted. So the deleted structures are the actual dummies that you have listed. A positive setting $n>0$ is useful if you have dummies pointing to other dummies, in a chain. Each dummy in the list is then substituted n times in order to determine which structure in each chain to delete.

Each time that DELETE is used, Genstat will also remove any unnamed structures that are no longer required and recover any space that has been used for temporary storage. This sort of tidying of workspace will happen automatically if Genstat sees in time that the space is becoming short. However, to avoid unnecessary computation, this does not occur after every statement. Thus, if the space appears to be exhausted, it may be worth using DELETE, even if you have no named structures to delete.

Options: REDEFINE, LIST, PROCEDURE, NSUBSTITUTE, REMOVE.
Parameter: unnamed.

See also

Genstat Reference Manual 1 Summary sections on: Data structures, Calculations and manipulation.

DEVICE

Switches between (high-resolution) graphics devices.

No options

Parameters

NUMBER = *scalar*	Device number
ENDACTION = *string token*	Action to be taken after completing each plot (continue, pause)
ORIENTATION = *string token*	Orientation of the pictures, if relevant (landscape, portrait); default * retains the current setting for this device
PALETTE = *string token*	How to represent colour (monotone, greyscale, grayscale, colour); default * retains the current setting for this device
RESOLUTION = *scalar*	Specifies the height of the image for hard-copy output, in pixels
ACTION = *string token*	How to create graphs for file types such as .emf, .jpg, .tif or .png (asynchronous, synchronous); default asyn

Description

High-resolution graphics can be generated principally in two forms by Genstat: either on a screen that can operate in graphics mode or by sending output to a file. The screen-based operation is for use in interactive sessions, whereas file output is designed for later use outside Genstat: either to produce hard-copy on a plotter or laser-printer, or to re-display graphics on the screen, if appropriate software is available. Usually there is a choice of various kinds of screen type or file format. Each type of output, whether screen or file, is referred to as a *device*; thus, the first step in producing graphical output is selecting a device within Genstat that is appropriate for the hardware that you have available. Genstat has built-in interfaces to several different graphics devices. These vary according to the Genstat implementation. However, a list of the devices and their associated numbers can be obtained using the DHELP procedure.

The output device is selected by the DEVICE statement. For example

```
DEVICE 4
```

selects the fourth available device.

If you have selected a file-based device you also need to open a file to receive the output, using the OPEN directive. This can be done before or after selecting the device, so long as the file has been opened before any output is generated. You can close the file when the graphics are complete; if you want to store separate items of graphical output in individual files you can use a sequence of OPEN and CLOSE statements. When opening or closing files for graphical output the CHANNEL parameter of the OPEN and CLOSE statements should be set to the device number specified by the DEVICE statement. For example:

```
OPEN 'Plot.jpg'; CHANNEL=7; FILETYPE=graphics
DEVICE 7
DGRAPH Y; X
CLOSE 7; FILETYPE=graphics
```

The default device, selected automatically when you start Genstat, is device 1: sometimes you may be able to specify an alternative device number and associated output file on the command line used to start Genstat (the local Genstat documentation should explain if this is possible).

You may get strange results if you try to generate graphics on a screen that is not designed for displaying graphics, or if you specify the wrong device type, as Genstat is not always able to

detect the type of device or screen.

There should be little difference in the use of Genstat graphics on different devices, as all the plotting symbols and character output are *software-generated* by default, using built-in graphics definitions and font files that are supplied with Genstat. The aspects of graphical output that may depend on particular capabilities of the graphics device are identified in Section 6.9 of Part 1 of the *Guide to the Genstat Command Language*; for example, different defaults may apply to colour and monochrome devices. It may sometimes be advantageous to use particular features of the hardware; for example, other fonts may be available. These device-specific features are usually selected by negative parameter settings (for example, `SYMBOL=-3`). Naturally, selection of device-specific attributes may lead to some differences in appearance of the output on different devices.

The `ENDACTION` parameter, with settings `continue` and `pause`, controls the action taken by default at the end of each plot. When using a graphics terminal interactively it may be convenient to pause at the end of a plot to examine the screen. When you are ready to continue, pressing carriage-return or some equivalent key will switch the terminal back to text mode and the Genstat prompt will appear. The `DKEEP` directive can provide the precise details for each particular device. For some interactive devices, for example PCs or workstations with separate graphics windows, it may not be necessary to pause. Each device is initialized to either `pause` or `continue` when you start Genstat, according to the particular implementation. If you are running in batch mode the default will always be to `continue`.

You can repeat the `DEVICE` statement and set `ENDACTION` to `pause` or `continue` at any time that you wish to change the default action. Alternatively, each graphical directive has an `ENDACTION` option that controls the device at the end of that directive, without altering the general default setting. For example, if you wish to build up a complex display using several `DGRAPH` statements with option `SCREEN=keep`, you could set `ENDACTION=continue` in the `DEVICE` statement, then put `ENDACTION=pause` in the final `DGRAPH` statement.

The `ORIENTATION` parameter can be used to specify `landscape` or `portrait` orientation of graphical output on PostScript and Interacter raster devices; `portrait` is the default. `PALETTE` can be set to `monotone`, to force all colours to be mapped to colour 1; this is the default for PostScript. Alternatively, `PALETTE=colour` produces colour PostScript output, and enables the use of the `COLOUR` directive to specify exactly the composition of the colours. The additional setting `PALETTE=greyscale` is as for monotone except that area filling (as in histograms) are shaded in grey tones, using the `RED` parameter of `COLOUR` to define the grey intensity.

The `RESOLUTION` parameter specifies the height of the image for hard-copy output, in pixels. (This is equivalent to setting the image resolution in the Options menu of the Genstat Graphics Viewer.)

The `ACTION` parameter controls how graphs are created for the file types `.emf`, `.jpg`, `.tif`, `.png`, `.gmf` and `.bmp`. The setting `synchronous` creates the graph before executing another command, whereas the setting `asynchronous` allows subsequent commands to be executed whilst the graph is created.

Options: none,
Parameters: `NUMBER`, `ENDACTION`, `ORIENTATION`, `PALETTE`, `RESOLUTION`, `ACTION`.

See also
Directives: `OPEN`, `CLOSE`, `DDISPLAY`, `DKEEP`.
Procedures: `DHELP`, `SETDEVICE`.
Genstat Reference Manual 1 Summary section on: Graphics.

DFINISH

Ends a sequence of related high-resolution plots.

No options or parameters

Description

This directive is relevant when a sequence of high-resolution graphics commands (such as DGRAPH and DHISTOGRAM) is to be used to build up a complicated plot. The start of the sequence is indicated using the DSTART directive. The information from each command is accumulated by Genstat, with no high-resolution plots being drawn until the DFINISH command is received.

Options: none.
Parameters: none.

See also

Directives: DSTART, DCLEAR.
Genstat Reference Manual 1 Summary section on: Graphics.

DFONT

Defines the default font for high-resolution graphics.

No options

Parameter

text	specifies or saves the default graphics font

Description

The Genstat Graphics Viewer can include textual information in a variety of fonts. A graphics pen can be asked to use a specific font family, by using the FONT parameter of the PEN directive. If this is not done, the pen is assumed to use the *default graphics font*. When Genstat is first installed, the default font is set automatically to Arial. However, it can be modified either by using menus in the Genstat Client or Graphics Viewer, or by using the DFONT directive.

DFONT has a single, unnamed, parameter, which can be set to text structure containing a single string. If that string is not missing (or null), it specifies the name of the font family to be used as the default. For example,

```
DFONT 'Calibri'
```

The name can be specified in upper or lower case, or in any mixture. You can find out the available fonts by looking at any of the controls for specifying fonts in the Client or Graphics Viewer.

If the text contains a missing string, it is redefined to contain the name of the font family currently used as the default. It is also defined as a text containing the current default, if you specify either a text with no values or an undeclared data structure.

Finally, if you specify DFONT without setting the parameter, it sets the default font back to the standard Genstat font i.e. Arial.

The change takes effect only when information about the new default is received by the Graphics Viewer. Afterwards this will be used in any graphs with the default font that are displayed or redisplayed, including those that have been stored in Genstat graphics meta files (i.e. files with the gmf suffix).

Options: none.
Parameter: unnamed.

See also

Directive: PEN.
Genstat Reference Manual 1 Summary section on: Graphics.

DGRAPH

Draws graphs on a plotter or graphics monitor.

Options

`TITLE` = *text*	General title; default `*`
`WINDOW` = *scalar*	Window number for the graphs; default 1
`KEYWINDOW` = *scalar*	Window number for the key (zero for no key); default 2
`SCREEN` = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (`clear`, `keep`, `resize`); default `clea`
`KEYDESCRIPTION` = *text*	Overall description for the key; default `*`
`ENDACTION` = *string token*	Action to be taken after completing the plot (`continue`, `pause`); default `*` uses the setting from the last `DEVICE` statement
`HOTMENU` = *matrices*	Defines sets of "hot" components for the user to select as shown or hidden by a menu in the Graphics Viewer
`HOTCHOICE` = *string token*	Whether one or several "hot" components can be displayed at a time (`one`, `several`); default `seve`

Parameters

`Y` = *identifiers*	Vertical coordinates
`X` = *identifiers*	Horizontal coordinates
`PEN` = *scalars*, *variates* or *factors*	Pen number for each graph (use of a variate or factor allows different pens to be defined for different sets of units); default `*` uses pens 1, 2, and so on for the successive graphs
`DESCRIPTION` = *texts*	Annotation for key
`YLOWER` = *identifiers*	Lower values for vertical bars
`YUPPER` = *identifiers*	Upper values for vertical bars
`XLOWER` = *identifiers*	Lower values for horizontal bars
`XUPPER` = *identifiers*	Upper values for horizontal bars
`YBARPEN` = *scalars*, *variates* or *factors*	Pens to use to draw the vertical bars; default −11
`XBARPEN` = *scalars, variates* or *factors*	Pens to use to draw the horizontal bars; default −11
`LAYER` = *scalars*	"Layer" of the plot
`UNITNUMBERS` = *identifiers*	Specifies unit numbers to be used when points are selected in the graphics viewer; default `*` uses the actual unit numbers of the values in the `X` and `Y` structures
`DISPLAY` = *string tokens*	Whether to display each component initially in the graph (`show`, `hide`); default `show`
`HOTCOMPONENT` = *scalars*	Allows components of the graph (specified by pairs of `Y` and `X` settings) to be defined as "hot" components that can be shown or hidden through their association with "hot" points or using a menu in the Graphics Viewer
`HOTDEFINITION` = *matrices*	Define how to use points defined by the Y and X parameters as "hot" points in the Graphics Viewer to allow the user to decide whether other components of the graph are shown or hidden

Description

The DGRAPH directive draws high-resolution graphs, containing points, lines or shaded polygons. The graph is produced on the current graphics device which can be selected using the DEVICE directive. The WINDOW option defines the window, within the plotting area, in which the graph is drawn; by default this is window 1.

The Y and X parameters specify the coordinates of the points to be plotted; they must be numerical structures (scalars, variates, factors, matrices or tables) of equal length. If any of the variates or factors is restricted, only the subset of values specified by the restriction will be included in the graph. The restrictions are applied to the Y and X variates or factors in pairs, and do not carry over to all the variates or factors in a list. For example, suppose the variate Y1 is restricted but the variate Y2 is not. The statement

```
DGRAPH Y1,Y2; X
```

will plot the subset of values of Y1 against X, but all the values of Y2 against X. Conversely, if X were restricted the subset would be plotted for both Y1 and Y2. Any associated structures, like variates specified by the PEN parameter or factors used to provide labels for the points, must be of the same length as Y and X.

Each pair of Y and X structures has an associated pen, specified by the PEN parameter. By default, pen 1 is used for the first pair, pen 2 for the second, and so on. The type of graph that is produced is determined by the METHOD setting of that pen. This can be point, to produce a point plot or scatterplot; line to join the points with straight lines; monotonic, open or closed to plot various types of curve through the points; or fill to produce shaded polygons. In the initial graphics environment, all the pens are defined to produce point plots. This can be modified using the METHOD option of the PEN directive. Other attributes of the pen can be used to control the colour, font, symbols and labels.

With METHOD=fill, the points defined by the Y and X variates are joined by straight lines to form one or more polygons which are then filled in the colours specified for the pen. The JOIN parameter of PEN determines the order in which the points are joined; with the default, ascending, the data are sorted into ascending order of x-values, while with JOIN=given they are left in their original order. There should be at least three points when using this method.

A warning message is printed if the data contain missing values. The effect of these depends on the type of graph being produced, as follows. If the method is point there will be no indication on the graph itself that any points were missing (but obviously none of the points with missing values for either the y- or x-coordinate can be included in the plot). If a line or curve is plotted through the points there will be a break wherever a missing value is found; that is, line segments will be omitted between points that are separated by missing values. When using METHOD=fill missing values will, in effect, define subsets of points, each of which will be shaded separately. Note, however, that the position of the missing values within the data will differ according to whether or not the data values have been sorted; this is controlled by the JOIN parameter of PEN, as described above. If the data are sorted, units with missing x-values are all moved to the beginning (and are thus ignored).

The PEN parameter can also be set to a variate or factor, to allow different pens to be used for different subsets of the units. With a factor, the units with each level are plotted separately, using the pen defined by the ordinal number of the level concerned. If PEN is set to a variate, its values similarly define the pen for each unit. For example, if you fit separate regression lines to some grouped data, you can easily plot the fitted lines in just two statements, one to set up the pens and one to plot the data:

```
PEN 1...Ngroups; METHOD=line; SYMBOL=0
DGRAPH Fitted; X; PEN=Groups
```

By default, Genstat calculates bounds on the axes that are wide enough to include all the data; the range of the data is extended by five percent at each end, and the axes are drawn on the left-hand side and bottom edge of the graph. This can all be changed by the XAXIS and YAXIS

directives using the LOWER and UPPER parameters to set the bounds, and YORIGIN and XORIGIN to control the position of the axes. Other parameters allow you to control the axis labelling and style. If the axis bounds are too narrow, some points may be excluded from the graph, so that *clipping* occurs. If the plotting method is point, Genstat ignores points that are out of bounds. For other settings of METHOD, lines are drawn from points that are within bounds towards points that are out of bounds, terminating at the appropriate edge. Clipping may also occur if the method is monotonic, open or closed and you have left Genstat to set default axis bounds, because these methods fit curves that may extend beyond the boundaries. If this occurs you should use the XAXIS and YAXIS directives to provide increased axis bounds. When you use several DGRAPH statements with SCREEN=keep to build up a complex graph, the axes are drawn only the first time, and the same axes bounds are then used for the subsequent graphs. You should then define axis limits that enclose all the subsequent data. Alternatively, if you set SCREEN=resize, the axes and their bounds will be adjusted, if necessary, to enclose the additional information. Axes are drawn only if SCREEN=clear, or the specified window has not been used since the screen was last cleared, or the window has been redefined by a FRAME statement.

DGRAPH allows *error bars* to be included in the plot. You might want to use these, for example, to show confidence limits on points that have been fitted by a regression. Error bars are requested by setting the YLOWER and YUPPER parameters to variates defining the lower and upper values for the error bar to be drawn at each point. For example, if you know the standard error for each point, you might calculate and plot the bounds as follows:

```
CALCULATE Barlow = Y - 1.96 * Err
& Barhigh = Y + 1.96 * Err
DGRAPH Y; X; YLOWER=Barlow; YUPPER=Barhigh
```

(this would give a 95% confidence interval assuming that the y-values come from a Normal distribution). The error bar is drawn from the lower point to the upper point at the associated x-position; the bar will be drawn even if the corresponding y-value (or y-variate) is missing. If the lower value is missing, or the YLOWER parameter is not set, only the upper section of the bar is drawn; likewise if the upper value is missing only the lower section is drawn. Similarly, parameters XLOWER and XUPPER allow you to plot horizontal bars at each point.

The YBARPEN and XBARPEN parameters define the pens to be used for the vertical and horizontal bars, respectively, with the default to use pen −11. Similarly to the PEN parameter, they can be set to either scalars, factors or variates. For each group of units defined by the setting of PEN, DGRAPH will use the first pen that it finds for that group in the setting supplied by YBARPEN and XBARPEN. (So YBARPEN and XBARPEN cannot define more detailed groupings of the points than those defined by PEN.) For example:

```
VARIATE [VALUES=1,1,2,2,3,3] Pvar
&       [VALUES=4,4,5,5,6,6] Ybvar
&       [VALUES=7,7,8,8,9,10] Xbvar
DGRAPH  Y; X; PEN=Pvar; YLOWER=Ylow; YUPPER=Yupp;\
        XLOWER=Xlow; XUPPER=Xupp;\
        YBARPEN=Ybpen; XBARPEN=Xbpen
```

The first two points here will be plotted in pen 1 with vertical bar in pen 4 and horizontal bar in pen 7. The third and fourth points will be plotted in pen 2 with vertical bar in pen 5 and horizontal bar in pen 8. The fifth and sixth points will be plotted in pen 3 with vertical bar in pen 6 and horizontal bar in pen 9. Notice, that the horizontal bar for the sixth point will be plotted in pen 9 *not* pen 10, as it is in the same PEN group as the (earlier) fifth point which has pen 9 for the horizontal bar. However, if PEN is not set to a factor or variate, the YBARPEN and XBARPEN settings define the groups.

The KEYWINDOW option specifies the window in which the key appears; by default this is window 2. Alternatively, you can set KEYWINDOW=0 to suppress the key. The key contains a line of information for each pair of Y and X structures, written with the associated pen. This will

indicate the symbol used, the line style (for a plotting method of `line` or `curve`) or a block to illustrate colour (when `METHOD=fill`), the name of the structure (if any) defined by the `LABELS` parameter of `PEN`, and a description indicating the identifiers of the data plotted (for example `Residuals v Fitted`). Alternatively, you can supply your own key, using the `DESCRIPTION` parameter, and you can specify a title for the key using the `KEYDESCRIPTION` option. If you draw several graphs using `SCREEN=keep` or `SCREEN=resize` and the same key window, each new set of information is appended to the existing key, until the window is full.

If you have set the `PEN` parameter to a variate or factor in order to plot independent subsets of the data, the key will contain information for each subset.
If the `LABELS` parameter of `PEN` has been used to specify labels for the points, each line of the key will contain the label corresponding to the first value of the subset, rather than the identifier of the labels structure itself.

The `TITLE` option can be used to provide a title for the graph. You can also put titles on the axes by using the `TITLE` parameter of the `XAXIS` and `YAXIS` directives. The `SCREEN` option controls whether the graphical display is cleared before the graph is plotted and the `ENDACTION` option controls whether Genstat pauses at the end of the plot.

The components of the graph defined by each pair of `Y` and `X` parameter settings are assumed to form separate, successive "layers" on the plot. So, if an area of the plot contains information (lines, symbols or labels) from several pairs of `Y` and `X` settings, the information from the later settings will overlay the information from earlier settings. You can control the orders of the layers by using the `LAYER` parameter to assign an explicit layer number to each pair of `Y` and `X` settings. The pairs of `Y` and `X` settings are then plotted in ascending order of layer numbers. These layer numbers also work across `DGRAPH` statements when you add to a plot by setting option `SCREEN=keep` or `SCREEN=resize`. So, for example, you can specify lower layer numbers to plot the new information "below" the layers formed by the earlier `DGRAPH` statement(s).

Usually all these components of the graph are shown when the graph is plotted. In Genstat *for Windows*, the Graphics Editor (which can be opened from the Edit menu on the menu bar of the Graphics Viewer) allows you to show or hide components, and the `DISPLAY` parameter of `DGRAPH` allows you to define whether a component should be shown or hidden in the initial graph displayed by the Graphics Viewer.

Alternatively, the Graphics Viewer itself can allow components to be shown or hidden, either by using their association with some "hot" points that have been defined on the graph, or by using a menu on its menu bar. These "hot" components are identified by defining a unique integer number for each one, using the `HOTCOMPONENT` parameter; if the component is not to be treated as "hot", `HOTCOMPONENT` should be left unset or given a missing value. Several pairs of `Y` and `X` parameter settings can be given the same number, so you can build up a "hot" component from more than one type of graphical item (e.g. from plotted points and shaded areas). "Hot" points are plotted within the graph using the `Y`, `X` and other parameters (e.g. `PEN`) in the usual way, as described above. The extra information, to define them as "hot", is supplied by setting the `HOTDEFINITION` parameter to a matrix with a row for each "hot" point, and a column for each type of "hot" component. The elements of the matrix specify the "hot" components to be associated with each "hot" point, using the numbers defined by the `HOTCOMPONENT` parameter. The menus in the Graphics Viewer can be made more informative, by defining textual labels for the rows and columns of the matrix (see the `MATRIX` directive); these are then used as annotation in the menus. Alternatively, if you set the `HOTMENU` option to a similar matrix, the Graphics Viewer will include a menu on its menu bar to allow users to choose whether "hot" components are shown or hidden. By default, users will be allowed to display several "hot" components at a time. However, you can set option `HOTCHOICE=one` to indicate that only one can be shown at a time. (The `DISPLAY` parameter should then be used to indicate which one, if any, should be shown on the initial graph.)

The Graphics Viewer also has a tool that allows you to select points, and copy their unit numbers onto the clipboard. Usually these numbers are simply the locations of the plotted values in the X and Y structures. However, you can use the UNITNUMBERS parameter to supply other numbers. (This may be useful if, for example, you are plotting sorted values.)

Options: TITLE, WINDOW, KEYWINDOW, SCREEN, KEYDESCRIPTION, ENDACTION, HOTMENU, HOTCHOICE.
Parameters: Y, X, PEN, DESCRIPTION, YLOWER, YUPPER, XLOWER, XUPPER, YBARPEN, XBARPEN, LAYER, UNITNUMBERS, DISPLAY, HOTCOMPONENT, HOTDEFINITION.

Action with RESTRICT
You can arrange to plot only a subset of the points specified by a particular pair of Y and X vectors and associated PEN vector, by restricting any one of them. If more than one of these is restricted, then they must all be restricted in exactly the same way.

See also
Directives: D3GRAPH, BARCHART, DHISTOGRAM, LPGRAPH, FRAME, XAXIS, YAXIS, AXIS, PEN.
Procedures: DCIRCULAR, DFUNCTION, DMSCATTER, DSCATTER, DTEXT, DFRTEXT, DXYDENSITY, TRELLIS.
Genstat Reference Manual 1 Summary section on: Graphics.

DHISTOGRAM

Draws histograms or bar charts on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the histograms; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
LIMITS = *variate*	Variate of group limits for classifying DATA variates into groups; default *
LOWER = *scalar*	For a DATA variate, this specifies the lower limit of the first bar; default * takes the minimum value of the variate
UPPER = *scalar*	For a DATA variate, this specifies the upper limit of the last bar; default * takes the maximum value of the variate
NGROUPS = *scalar*	When LIMITS and BINWIDTH are not specified, this defines the number of groups into which a DATA variate is to be classified; default is then 10, or the integer value nearest to the square root of the number of values in the variate if that is smaller
BINWIDTH = *scalar*	When LIMITS is unset the range of a DATA variate is split into equal intervals known as "bins" to form the groups, this option can set the bin widths (alternative is to set the number of groups using NGROUPS)
FIXEDBARWIDTH = *string token*	Whether to plot the histogram with bars of equal width (yes, no); default no
BARCOVERING = *scalar*	What proportion of the space allocated along the x-axis each bar should occupy; default * gives proportion 1 for a DATA variate, and 0.8 for a factor or table (thus giving a gap between each bar)
BARSCALE = *scalar*	Width of bar for which one unit of bar length represents one unit of data; default * uses the width of the narrowest bar
LABELS = *text*	Group labels; default *
APPEND = *string token*	Whether or not the bars of the histograms are appended together (yes, no); default no
ORIENTATION = *string token*	Direction of the plot (horizontal, vertical); default vert
OUTLINE = *string token*	Where to draw outlines (bars, perimeter); default bars
PENOUTLINE = *scalar*	Pen to use for the outlines; default -8
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

DATA = *identifiers*	Data for the histograms; these can be either a factor

	indicating the group to which each unit belongs, a variate whose values are to be grouped, or a one-way table giving the height of each bar
NOBSERVATIONS = *tables*	One-way table to save numbers in the groups
GROUPS = *factors*	Factor to save groups defined from a variate
PEN = *scalars* or *variates*	Pen number(s) for each histogram; default * uses pens 2, 3, and so on for the successive structures specified by DATA
DESCRIPTION = *texts*	Annotation for key

Description

DHISTOGRAM plots high-resolution histograms or bar charts, depending on the input supplied by the DATA parameter: either a list of variates, a list of factors or a list of one-way tables. For a DATA variate, a histogram is produced. This summarizes the distribution of the variate by counting the number of values within a set of intervals defined by the LIMITS, NGROUPS or BINWIDTH options. The histogram contains a "bar" for each interval, with area proportional to the number of values found there. You can define the boundaries between each interval using the LIMITS option. Alternatively, instead of setting LIMITS, you can specify the width of each interval using the BINWIDTH option. Or, instead of setting LIMITS or BINWIDTH, you can specify the number of groups using the NGROUPS option. Finally, if none of these options is set, Genstat defines the number of groups to be 10, or the integer value nearest to the square root of the number of values in the first DATA variate if that is smaller. The range of the histogram is specified by the LOWER and UPPER options. LOWER defines the lower limit of the first interval; by default this is set by making the width of the first bar equal to the width of the second bar, or it is the minimum value of the variates if that would otherwise be below the first bar. UPPER defines the upper limit of the last interval; by default this is set by making the width of the final bar equal to the width of the last-but-one bar, or it is the maximum value of the variates if that would otherwise be above the final bar. The bars are perpendicular to the x-axis, and this is labelled with the positions of the interval boundaries.

If you set DATA to factors or tables, bar charts are produced. However, these can be plotted more conveniently by the BARCHART directive. Bar charts differ from histograms in that there is no longer the concept of dividing the x-axis into a set of contiguous intervals. Instead we have a set of bars located at various positions along the x-axis. The bars are spaced equally along the x-axis. If DATA is set to a list of factors, the bars are labelled by the labels, if available, or otherwise the levels of the first factor. If DATA is set to a list of tables, the labelling is given by the levels/labels of the factor classifying the first table. A DATA table defines the heights of each bars directly (from the value in the corresponding cell of the table). With a factor, Genstat first constructs a table giving the replications of the factor levels. So the height of each bar is equal to the number of units of the factor with the corresponding level of the factor.

The bars in a bar chart always have equal widths. With a histogram, the default is for the bar widths to be equal to the widths of the underlying intervals. However, you can request equal bar widths by setting option FIXEDBARWIDTH=yes. The BARCOVERING option indicates what proportion of the space allocated along the x-axis each bar should occupy. For a histogram the default is 1, while for bar charts it is 0.8 (thus giving a gap between each bar).

The BARSCALE option controls how the lengths of the bars correspond to units of data. The length of each bar is calculated as (data-value × BARSCALE)/bar-width. By default, BARSCALE is set to the width of the narrowest bar. So for that bar, the length will correspond directly to the data units.

The WINDOW option defines the window where the histogram is plotted, and the KEYWINDOW option similarly specifies where the key should appear. You can set either of these to zero if you want to suppress the corresponding output. Titles can be added to the histogram and key using

the TITLE and KEYDESCRIPTION options respectively.

The APPEND option controls the form of display to be used when the DATA parameter specifies a list of structures. These parallel histograms can be produced in one of two styles. By default (APPEND=no), the histogram contains a set of bars for each structure, drawn in parallel groups. Alternatively, if you set APPEND=yes, the bars for the structures are concatenated into a single bar for each group. The bottom portion of each bar then corresponds to the first structure, and the top to the last structure.

The ORIENTATION option controls whether the bars of the histogram are plotted vertically (the default) or horizontally. When ORIENTATION=horizontal, the horizontal axis is taken to be the y-axis, so the same XAXIS and YAXIS settings can be used however the histogram is oriented.

The bars for each structure are all shaded according to the pen or pens that have been specified for that structure, using the PEN parameter. You can set PEN to a scalar to define a single pen to be used for all the bars, or to a variate to define a different pen for each bar. If PEN is not set, Genstat uses the pens in turn, pen 2 for the first structure, pen 3 for the second structure, and so on, so that a different shading is used for each structure. The relevant aspects of the pens should be set in advance, if required, using the COLOUR parameters of the PEN directive. Generally, however, the default attributes of the pens will be satisfactory.

The OUTLINE option controls whether lines are drawn around the bars or around the perimeter of the histogram. These are drawn using the pen specified by the PENOUTLINE option (default −8). You can suppress all the outlines by setting OUTLINE=*.

The axes of the histogram are formed automatically from the data. By default, the upper bound of the y-axis is set to be five percent greater than the height of the longest bar. If any of the bars has a negative height the lower bound is adjusted in a similar way, otherwise it is set to zero. As already mentioned, when the histogram is formed from a variate, the x-axis markings are set to indicate the limits of each bar or set of bars; when the data are provided in a factor the factor labels or levels are used to label the histogram bars, and when the bar heights are provided directly in a table the classifying factor of the table is used. You can control the form of the axes by using the XAXIS and YAXIS directives to set the required attributes before the DHISTOGRAM directive is used.

The WINDOW parameter of XAXIS and YAXIS should be set to the window in which the histogram is to be plotted (controlled by the WINDOW option of DHISTOGRAM). The TITLE, LOWER, UPPER, MARKS and LABELS parameters control annotation. The UPPER parameter of YAXIS is particularly useful when you are plotting a series of histograms; by setting UPPER to a value larger than any of the bars in any of the histograms, you can ensure that they are all plotted on the same scale.

The histogram key consists of the title, if set by KEYDESCRIPTION, followed by a legend for each structure plotted. This consists of a small rectangle that is drawn in the same colour as that used in the histogram, followed by the identifier name or the piece of text specified by the DESCRIPTION parameter.

The SCREEN option controls whether the graphical display is cleared before the histogram is plotted and the ENDACTION option controls whether Genstat pauses at the end of the plot.

Options: TITLE, WINDOW, KEYWINDOW, LIMITS, LOWER, UPPER, NGROUPS, BINWIDTH, FIXEDBARWIDTH, BARCOVERING, BARSCALE, LABELS, APPEND, ORIENTATION, OUTLINE, PENOUTLINE, SCREEN, KEYDESCRIPTION, ENDACTION.
Parameters: DATA, NOBSERVATIONS, GROUPS, PEN, DESCRIPTION.

Action with RESTRICT

You can restrict a DATA variate or factor to form a histogram for only a subset of the units. However, the restriction does not carry over to any other variates or factors listed by the DATA

parameter.

See also
Directives: BARCHART, D3HISTOGRAM, DPIE, LPHISTOGRAM, FRAME, XAXIS, YAXIS, PEN.
Procedures: TRELLIS, DOTHISTOGRAM, DOTPLOT, DCIRCULAR, WINDROSE.
Genstat Reference Manual 1 Summary section on: Graphics.

DIAGONALMATRIX

Declares one or more diagonal matrix data structures.

Options

ROWS = *scalar*, *vector*, *pointer* or *text*	Number of rows, or labels for rows (and columns); default *
VALUES = *numbers*	Values for all the diagonal matrices; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the diagonal matrices in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the diagonal matrices
VALUES = *identifiers*	Values for each diagonal matrix
DECIMALS = *scalars*	Number of decimal places for printing
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for the contents of each structure
MAXIMUM = *scalars*	Maximum value for the contents of each structure
DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represent dates and times

Description

Diagonal matrices are square matrices that have zero entries except on their leading diagonals: for example,

```
2  0  0
0  1  0
0  0  3
```

Another example is the identity matrix, which has a diagonal of values equal to 1. To save space, Genstat has a special structure for diagonal matrices, and these can be declared using the DIAGONALMATRIX directive.

Because a diagonal matrix is square, Genstat requires you to specify only the number of rows. This is done using the ROWS option. The simplest method is to use a scalar to define the number of rows explicitly. Alternatively, you can set ROWS to a variate, text or pointer, whose length then defines the number of rows and whose values will then be used as labels, for example when the matrix is printed. Finally, if you specify a factor, the number of levels defines the number of rows and the labels if available, or otherwise the levels, are used for labelling.

When you give the values of a diagonal matrix, either in a declaration or when its values are read, you should specify only the diagonal elements. (Genstat does not store the off-diagonal elements, but assumes them to be zero.) Similarly, when a diagonal matrix is printed it appears as a column of numbers; Genstat omits the off-diagonal zeros. For example:

```
DIAGONALMATRIX [ROWS=3; VALUES=2,1,3] D
```

declares the diagonal matrix D and gives it the values shown above.

Values can be assigned to the diagonal matrices by either the VALUES option or the VALUES parameter. The option defines a common value for all the matrices in the declaration, while the parameter allows them each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

If the MODIFY option is set to yes any existing attributes and values of the diagonal matrices are retained (if still appropriate); otherwise these are lost.

The DECIMALS parameter allows you to define a number of decimal places to be used by default when each diagonal matrix is printed. You can associate a text of extra annotation with each diagonal matrix using the EXTRA parameter. The MINIMUM and MAXIMUM parameters allow you to define lower and upper limits on the values in each diagonal matrix. Genstat then prints warnings if any values outside that range are allocated to the matrix. The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each diagonal matrix to indicate that the matrix stores dates and times, and to define a format to be used for these, by default, when they are printed; details are given in the description of the PRINT directive.

The IPRINT option can be set to specify how the diagonal matrices will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

Options: ROWS, VALUES, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUES, DECIMALS, EXTRA, MINIMUM, MAXIMUM, DREPRESENTATION.

See also
Directives: LRV, MATRIX, SYMMETRICMATRIX, SSPM.
Genstat Reference Manual 1 Summary section on: Data structures.

DISPLAY

Prints, or reprints, diagnostic messages.

Options

PRINT = *string token*	What information to print (diagnostic); default diag
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
FAULT = *text*	Specifies the fault message to print (for example, FAULT='VA 4' prints the message "Values not set"); default is to print the last diagnostic message

No parameters

Description

By default, DISPLAY reprints the most recent diagnostic. Alternatively, you can use the FAULT option of DISPLAY to print any particular Genstat diagnostic. The CHANNEL option controls where the information is printed; the default is the current output file.

Options: PRINT, CHANNEL, FAULT.
Parameters: none.

See also

Directives: FAULT, GET.
Genstat Reference Manual 1 Summary section on: Program control.

DISTRIBUTION

Estimates the parameters of continuous and discrete distributions.

Options

PRINT = *string tokens*	Printed output required from each individual fit (parameters, samplestatistics, fittedvalues, proportions, monitoring); default para, samp, fitt
CBPRINT = *string tokens*	Printed output required from a fit combining all the input data (parameters, samplestatistics, fittedvalues, proportions, monitoring); default *
DISTRIBUTION = *string token*	Distribution to be fitted (Poisson, geometric, logseries, negativebinomial, NeymanA, PolyaAeppli, PlogNormal, PPascal, Normal, dNvequal, dNvunequal, logNormal, exponential, gamma, Weibull, b1, b2, Pareto); default * i.e. fit nothing
CONSTANT = *string token*	Whether to estimate a location parameter for the gamma, logNormal, Pareto or Weibull distributions (estimate, omit); default omit
LIMITS = *variate*	Variate to specify or save upper limits for classifying the data into groups; default *
NGROUPS = *scalar*	When LIMITS is not specified, this defines the number of groups (of approximately equal size) into which the data are to be classified; default is the integer value nearest to the square root of the number of data values
XDEVIATES = *variate*	Variate to specify points up to which the CUMPROPORTIONS are to be estimated
JOINT = *string token*	Requests joint estimates from the combined fit to be used for a re-fit to the separate data sets (dispersion, variancemeanratio, Poissonindex); default *
PARAMETERS = *variate*	Estimated parameters from the combined fit
SE = *variate*	Standard errors for the estimated parameters of the combined fit
VCOVARIANCE = *symmetric matrix*	Variance-covariance matrix for the estimated parameters of the combined fit
CUMPROPORTIONS = *variate*	Estimated cumulative proportions of the combined distribution up to the values specified by the XDEVIATES option
MAXCYCLE = *scalar*	Maximum number of iterations; default 30
TOLERANCE = *scalar*	Convergence criterion; default 0.0001

Parameters

DATA = *variates* or *tables*	Data values either classified (table) or unclassified (variate)
NOBSERVATIONS = *tables*	One-way table to save the data classified into groups
RESIDUALS = *tables*	Residuals from each (individual) fit
FITTEDVALUES = *tables*	Fitted values from each fit
PARAMETERS = *variates*	Estimated parameters from each fit
SE = *variates*	Standard errors of the estimates

VCOVARIANCE = *symmetric matrices*	Variance-covariance matrix for each set of estimated parameters
CUMPROPORTIONS = *variates*	Estimated cumulative proportions of each distribution up to the values specified by the XDEVIATES option
CBRESIDUALS = *tables*	Residuals from the combined fit
CBFITTEDVALUES = *tables*	Fitted values from the combined fit
STEPLENGTH = *variates*	Initial step lengths for each fit
INITIAL = *variates*	Initial values for each set fit

Description

The DISTRIBUTION directive is used to fit an observed sample of data to a theoretical distribution function, in order to obtain maximum-likelihood estimates of the parameters of the distribution and test the goodness of fit. The data consists of observations x_i of a random variable X, which has a distribution function $F(x)$ defined by $F(x)=\Pr(X \le x)$. A selection of both discrete and continuous distributions are available; full details are given below.

For discrete distributions X may take non-negative integer values only, except for the log-series distribution where only positive integer values are allowed. For continuous distributions the random variable X may take any values, subject to constraints for certain distributions, for example, data values must be strictly positive in order to fit a log-Normal distribution. Constraints are detailed with the individual distributions described below.

The data can be supplied to DISTRIBUTION as a variate or as a one-way table of counts. If the raw data are available, then these should be supplied (as a variate), since the raw data contains more information than grouped data.

If raw data are not available, then a one-way table of counts, or frequencies, should be given. The factor classifying the table must have its levels vector declared explicitly, since the levels are used to indicate the boundary values of the raw data used to create the grouping. For example, if the discrete variable X takes the values 0...8, with numbers of observations 2,6,7,4,2,1,0,1,0 respectively, a table of counts can be declared by

```
FACTOR [LEVELS=!(0...8)] F
TABLE [CLASSIFICATION=F; VALUES=2,6,7,4,2,1,0,1,0] T
```

The factor levels do not have to specify single data values: often it will be desirable to group certain values together, and indeed for continuous data this is the only sensible way to proceed. In general, for a classifying factor with levels $l_1, l_2, \ldots, l_f$, the count n_k for the kth cell of the table will be the number of observations x_i such that

$x_i \le l_1$,	$k=1$
$l_{k-1} < x_i \le l_k$,	$2 \le k \le f-1$
$l_{f-1} < x_i$,	$k=f$

This means that for all except the last cell of the table, the factor level represents the upper limit on values in that cell. The final class of the table is termed the *tail*; it is formed by combining the frequencies for all values of X greater than l_{f-1}, and the upper limit on values in the tail is infinity. For continuous distributions with no lower bound, the first class will be the lower tail. You will often want to form the tail(s) by amalgamating groups with low numbers of counts. In the example above, you might amalgamate the groups for values 6-8:

```
FACTOR [LEVELS=!(0...5,99)] F2
TABLE [CLASSIFICATION=F2; VALUES=2,6,7,4,2,1,1] T2
```

Note that the final factor level, for the tail, can be given a dummy value of 99 to indicate that it has no upper limit, since this value is never used in calculations.

When data are supplied as a table instead of as a variate, the computed log-likelihood is only an approximation to the full log-likelihood and the solution obtained will depend to some extent

on the choice of class limits. More reliable results will be achieved with a larger number of classes, since this gives more information on the data distribution, so only classes with very few observations should be amalgamated. In general, care should be taken to choose class limits that give a reasonable number of counts in each class, but with none of the individual classes holding a disproportionately large number of observations.

The `DISTRIBUTION` option should be set to indicate which distribution is to be fitted to the data. The following distributions are available:

Discrete	**Continuous**
Binomial (as a special case of the negative binomial)	Normal
Poisson	Double Normal (equal variances)
Geometric	Double Normal (unequal variances)
Log-series	Log-Normal
Negative binomial	Exponential
Neyman type A	Gamma
Pólya-Aeppli	Weibull
Poisson-log-Normal	Beta type I and type II
Poisson-Pascal	Pareto

Note: the parameterization for the gamma distribution differs from that used in the gamma probability functions. `DISTRIBUTION` uses the *shape* parameter k and the *rate* parameter b, while the functions use the *shape* parameter k and the *scale* parameter t, which is the reciprocal of the rate ($t=1/b$).

The first step of the fitting process is to compute and print various sample statistics. Examining these may help in the selection of appropriate distributions for fitting – properties of the various distributions are listed at the end of this section. The setting `DISTRIBUTION=*` can be used to produce this output without any model fitting. The following sample statistics are calculated:

Sample size	n	
Sample mean	$m = \Sigma\, x_i/n$	
Sample variance	$s^2 = \Sigma\, x_i^2/n - m^2$	discrete distributions
	$s^2 = \Sigma\, (x_i-m)^2 / (n-1)$	continuous distributions
Sample skewness	$g_1 = \Sigma\, (x_i-m)^3 / (n-1)s^3$ $= m_3/s^3$x	
Sample kurtosis	$g_2 = \Sigma\{(x_i-m)^4/(n-1)s^4\} - 3$	continuous distributions only
Sample quartiles	x_p: $F(x_p)=p$	
Poisson index	$(s^2-m)/m^2$	discrete distributions only
Negative binomial index	$m(m_3-3s^2+2m)/(s^2-m)^2$	discrete distributions only

If the original data are not available, the sample statistics are calculated by substituting class mid-points in place of the data. For the lower tail, the class "mid-point" is taken to be $l_1-\frac{1}{2}(l_2-l_1)$ and for the upper tail, $l_{f-1}+\frac{1}{2}(l_{f-1}-l_{f-2})$. No corrections are made for groupings. When a distribution has been fitted to data, the relevant theoretical statistics of that distribution are printed for comparison with the sample statistics, as a check on the appropriateness of the model for the data.

A summary is given of the fit: the parameter estimates are printed with their standard errors and correlations, including the *working parameters*, which are *stable* functions of the parameters defining the distribution and are used in the internal algorithm. The goodness of fit to the chosen distribution is indicated by the residual deviance which has an asymptotic chi-square distribution with the specified degrees of freedom. The deviance is also the preferred statistic for comparison

of nested models, for example the double Normal distribution with equal and unequal variances. This is followed by a table of observed and fitted values (expected frequencies), together with weighted residuals. If raw data are supplied, by default this table is formed by dividing the data into $\sqrt{n}$ groups of approximately equal observed frequency, which are therefore likely to be of unequal widths. The NGROUPS option may be used to set the number of groups for this table. If data are supplied as a table, the fitted values use the classification from that table. In either case the LIMITS option may be used to supply a different set of limits; with the constraint that if tabulated data are analysed these limits should be a subset of the original limits so that the new groups are formed by aggregation.

The NOBSERVATIONS, RESIDUALS and FITTEDVALUES parameters can be used to save the number of observations in each cell, the fitted number, and the residual respectively (all in tables). The parameter estimates and their standard errors can be saved in variates specified by PARAMETERS and SE. The variance-covariance matrix for the estimated parameters can be saved as a symmetric matrix using the VCOVARIANCE parameter.

Having fitted the required distribution, the estimated cumulative distribution function (CDF) can be evaluated at specified values of *X*. These are defined using the XDEVIATES option. The values of the CDF can be printed (by selecting PRINT=proportions) or saved in a variate by setting the CUMPROPORTION parameter.

If you have several sets of data you may be interested in fitting the distribution individually to each set; this can be done by setting the DATA parameter to a list of identifiers. A separate analysis is then performed for each set of data, but of course any option settings are common to all the data sets. The data sets should all be specified in the same way, either as raw data or as tabulated counts. For tabulated counts, the same categories must be used for defining every table. You can also carry out one final fit to the combined data set, in order to investigate whether the data can be adequately modelled as coming from a single population. This combined fit is produced if any of the options relating to the combined fit have been set (that is, options CBPRINT, PARAMETERS, SE, VCOVARIANCE or CUMPROPORTION which print or save information from the combined analysis). For each individual data set you can also save fitted values and residuals based on the parameters estimated from the combined data set, using the CBRESIDUALS and CBFITTEDVALUES parameters. The JOINT option can be used to specify that certain parameters should be held constant at their estimated values from the combined analysis during refits to the individual data sets. For continuous distributions only, a common dispersion parameter can be requested; for discrete distributions a common value can be requested for either the Poisson index or the ratio of variance to mean. An analysis of deviance is printed to compare the nested models.

If the original data are available, the full log-likelihood is used in the optimization algorithm. Otherwise, an approximate log-likelihood is optimized, using representative values for each class. For some distributions, it is necessary to use stable *working parameters* in the optimization algorithm (Ross 1990), and the *defining parameters* for the distribution are then evaluated by a simple transformation.

The deviance and corresponding degrees of freedom that are printed as part of the model summary are based on the table of fitted values, and thus may be affected by the choice of limits. The residuals computed are deviance residuals (McCullagh & Nelder 1989), and the deviance is therefore the sum of squared residuals. The degrees of freedom are $n-p-1$, where n is the number of cells in the table of fitted values and p is the number of parameters estimated in the model. The default limits for grouping the raw data are designed to avoid small expected frequencies (for example in the tail cells) which can have an inflationary affect on the deviance; however, if the tails are important, because of the origin of the data, it may be important to specify the limits explicitly.

An iterative Gauss-Newton optimization method is used to estimate the parameters of the distribution. The parameterization is chosen for each model so that the optimization is stable, but

if there are any problems with particular data sets it may be necessary to control this process. The MAXCYCLE and TOLERANCE options allow you to increase the number of iterations and alter the convergence criterion for data sets that fail to converge. You can also specify initial values and step lengths for the parameters for each set of data using the STEPLENGTH and INITIAL parameters. These parameters should be set to variates of length appropriate for the distribution being fitted; for example, if DISTRIBUTION=Poisson they should have just one value. Another use of INITIAL and STEPLENGTH is to constrain a parameter to a particular value; for example when fitting a double Normal the proportion parameter *p* could be fixed at 0.5 by setting the initial value to 0.5 and the step length to 0, thus fitting a double Normal in equal proportions. Note that the degrees of freedom are not adjusted to take account of this.

Options: PRINT, CBPRINT, DISTRIBUTION, CONSTANT, LIMITS, NGROUPS, XDEVIATES, JOINT, PARAMETERS , SE, VCOVARIANCE, CUMPROPORTIONS, MAXCYCLE, TOLERANCE.
Parameters: DATA, NOBSERVATIONS, RESIDUALS, FITTEDVALUES, PARAMETERS, SE, VCOVARIANCE, CUMPROPORTIONS, CBRESIDUALS, CBFITTEDVALUES, STEPLENGTH, INITIAL.

Action with RESTRICT
You can restrict the units of a DATA variate to fit a distribution to a subset of its values.

References
McCullagh, P. & Nelder, J.A. (1989). *Generalized Linear Models* (second edition). Chapman and Hall, London.
Ross, G.J.S. (1990). *Nonlinear Estimation*. Springer-Verlag, New York.

See also
Procedures: BBINOMIAL, CUMDISTRIBUTION, DPROBABILITY, EDFTEST, FDRMIXTURE, KERNELDENSITY, NORMTEST, WSTATISTIC, RSURVIVAL.
Functions: CLBETA, CLBINOMIAL, CLBVARIATENORMAL, CLCHISQUARE, CLF, CLGAMMA, CLHYPERGEOMETRIC, CLINVNORMAL, CLLOGNORMAL, CLNORMAL, CLOGLOG, CLPOISSON, CLSMMODULUS, CLSRANGE, CLT, CLUNIFORM, CUBETA, CUBINOMIAL, CUBVARIATENORMAL, CUCHISQUARE, CUF, CUGAMMA, CUHYPERGEOMETRIC, CUINVNORMAL, CULOGNORMAL, CUNORMAL, CUPOISSON, CUSMMODULUS, CUSRANGE, CUT, CUUNIFORM, EDBETA, EDBINOMIAL, EDCHISQUARE, EDF, EDGAMMA, EDHYPERGEOMETRIC, EDINVNORMAL, EDLOGNORMAL, EDNORMAL, EDPOISSON, EDSMMODULUS, EDSRANGE, EDT, EDUNIFORM, GRBETA, GRBINOMIAL, GRCHISQUARE, GRF, GRGAMMA, GRHYPERGEOMETRIC, GRLOGNORMAL, GRNORMAL, GRPOISSON, GRSAMPLE, GRSELECT, GRT, GRUNIFORM, PRBETA, PRBINOMIAL, PRCHISQUARE, PRF, PRGAMMA, PRHYPERGEOMETRIC, PRINVNORMAL, PRLOGNORMAL, PRNORMAL, PRPOISSON, PRSMMODULUS, PRSRANGE, PRT, PRUNIFORM.
Genstat Reference Manual 1 Summary section on: Basic and nonparametric statistics.

DKEEP

Saves information from the last plot on a particular device.

No options

Parameters

DEVICE = *scalars*	The devices for which information is required, if the scalar is undefined or contains a missing value, this returns the current device number
WINDOW = *scalars*	Window about which the information is required; default * gives information about the last window
XLOWER = *scalars*	Lower bound for the x-axis in last graph in the specified device and window
XUPPER = *scalars*	Upper bound for the x-axis in last graph in the specified device and window
YLOWER = *scalars*	Lower bound for the y-axis in last graph in the specified device and window
YUPPER = *scalars*	Upper bound for the y-axis in last graph in the specified device and window
ZLOWER = *scalars*	Lower bound for the z-axis in last graph in the specified device and window
ZUPPER = *scalars*	Upper bound for the z-axis in last graph in the specified device and window
FILE = *scalars*	Returns the value 1 or 0 to indicate whether a file is required for this device
DESCRIPTION = *texts*	Description of the device
DREAD = *scalars*	Returns the value 1 or 0 to indicate whether graphical input is possible from this device
ENDACTION = *texts*	Returns the current ENDACTION setting ('continue' or 'pause')

Description

DKEEP provides information that can be used in general programs and procedures to control the graphical output. For the specified device you can determine whether it generates screen output or uses a file, whether graphical input is possible, a description of the device, the current ENDACTION setting, and details of the axis bounds.

The device for which the information is required is specified by the DEVICE parameter. If you specify a scalar containing a missing value, this will be set to the number of the current graphics device. You can then test whether an output file is needed and open one accordingly.

When writing a procedure you can find out if axes bounds have been set explicitly, using the SAVE parameter of XAXIS, YAXIS and ZAXIS. This information may then be used when setting up the axes for other graphs. However, if the bounds were not set, but have been evaluated from the data (or if the axes have subsequently been redefined) the information in the save structure will not be of any use. The actual values used when plotting are recorded internally, for each window of each device, and can be accessed using the XLOWER, XUPPER, YLOWER, YUPPER, ZLOWER and ZUPPER, parameters of DKEEP.

Options: none.

Parameters: DEVICE, WINDOW, XLOWER, XUPPER, YLOWER, YUPPER, ZLOWER, ZUPPER, FILE, DESCRIPTION, DREAD, ENDACTION.

See also
Directives: DEVICE, DLOAD, DSAVE, XAXIS, YAXIS, ZAXIS, PEN.
Procedure: DHELP.
Genstat Reference Manual 1 Summary section on: Graphics.

DLOAD

Loads the graphics environment settings from an external file.

No options

Parameter

text	File from which to lead the environment settings

Description

DLOAD allows you to (re)load the graphics environment settings into Genstat, from an external file saved earlier by the DSAVE directive. If the parameter is unset, Genstat restores the default graphics environment.

Options: none.
Parameter: unnamed.

See also

Directives: DSAVE, FRAME, XAXIS, YAXIS, ZAXIS, PEN, DEVICE, COLOUR.
Procedures: DHELP, FFRAME, GETRGB.
Genstat Reference Manual 1 Summary section on: Graphics.

DPIE

Draws a pie chart on a plotter or graphics monitor.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the pie chart; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ANNOTATION = *string token*	How to annotate each slice (description, percentage); default desc, perc
OUTLINE = *string token*	Where to draw outlines (slices, perimeter); default slices
PENOUTLINE = *scalar*	Pen to use for the outlines; default -10
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

SLICE = *scalars*	Amounts in each of the slices (or categories)
PEN = *scalars*	Pen number for each slice; default * uses pens 1, 2, and so on for the successive slices
DESCRIPTION = *texts*	Description of each slice

Description

A pie chart is formed by taking the values of the scalars in the SLICE parameter, in order, and representing them by segments of a circle starting at "three o'clock" and working in an anti-clockwise direction. The angle subtended by each segment (and thus the area of the segment) is proportional to the value of the corresponding scalar. The values may be raw data or can be expressed as percentages (by ensuring they total 100).

The colour used for each segment can be controlled using the PEN parameter. By default, pen 1 is used for the first segment, pen 2 for the second segment, and so on. The default colours differ from pen to pen, and can be modified using the PEN directive.

Individual segments can be displaced outwards from the centre, to obtain an "exploded" pie chart. The chosen segments are indicated by setting the corresponding scalars in the SLICE parameter list to negative values.

The WINDOW and KEYWINDOW options specify the windows in which the pie chart and key are to be displayed. The shape of the pie chart is determined by the dimensions of the window; if it is not square the resulting pie chart will be elliptical.

Titles can be added using the TITLE and KEYDESCRIPTION options. The key produced for the pie chart is similar to that produced by the DHISTOGRAM directive. A shaded block is drawn for each segment, followed by the annotation requested using the settings of the ANNOTATION option:

description	the text supplied by the DESCRIPTION parameter or, if this is not set, the identifier of the SLICE scalar;
percentage	the percentage contained in the slice.

The OUTLINE option controls whether lines are drawn around the slices or around the perimeter of the pie chart. These are drawn using the pen specified by the PENOUTLINE option (default -10). You can suppress all the outlines by setting OUTLINE=*.

The SCREEN option controls whether the graphical display is cleared before the histogram is plotted and the ENDACTION option controls whether Genstat pauses at the end of the plot.

Options: TITLE, WINDOW, KEYWINDOW, ANNOTATION, OUTLINE, PENOUTLINE, SCREEN, KEYDESCRIPTION, ENDACTION.
Parameters: SLICE, PEN, DESCRIPTION.

See also
Directives: BARCHART, DHISTOGRAM, D3HISTOGRAM, LPHISTOGRAM, FRAME, XAXIS, YAXIS, PEN.
Procedures: TRELLIS, DOTHISTOGRAM, DOTPLOT, DCIRCULAR, WINDROSE.
Genstat Reference Manual 1 Summary section on: Graphics.

DREAD

Reads the locations of points from an interactive graphical device.

Options

PRINT = *string tokens*	What to print (data, summary); default summ
CHANNEL = *scalar*	Number of the graphics device from which to read; default * takes the current graphics device
WINDOW = *scalar*	Window from which to read; default 1
CURSORTYPE = *scalar*	Type of cursor; default 1
SETNVALUES = *string token*	Whether to set number of values of structures from the number of values read (yes, no); default no causes the number of values to be set only for structures whose lengths are not defined already
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

Y = *variates*	Variate to receive the y-values that have been read
X = *variates*	Variate to receive the x-values that have been read
YGIVEN = *variates*	Y-coordinates of points that may be located on the graph
XGIVEN = *variates*	X-coordinates of points that may be located
SAVESET = *variates*	Unit numbers of the located points
PEN = *scalars*	Pen number to use to echo points; default 0
YSAVE = *variates*	Variate to receive the y-coordinates of the located points
XSAVE = *variates*	Variate to receive the x-coordinates of the located points

Description

The DREAD directive allows you to input information about the positions of points on interactive graphical terminals. The exact details of how this directive operates will vary slightly from one system to another, so this section attempts to outline the basic principles involved. If you encounter any difficulties using DREAD you should refer to the *Users' Note* supplied with your version of Genstat.

When you type DREAD, a cursor should appear on the graphics screen. This can be moved to the chosen position by using the cursor keys or a mouse; the coordinates of this point can then be read by pressing a key or mouse button (normally the left hand mouse button). The cursor can then be moved to another position to read the next point. You can use graphical input within any window that contains a graph or contour plot, but you cannot input data from an "empty" window or one containing other forms of graphical output. In addition you can identify particular points from those plotted on an existing graph and you can mark the points that you have read.

The CHANNEL and WINDOW options are used to specify the device and the window from which the information is to be read; the default is to read from window 1 of the current device. The values that are read are converted to the scale of the data that was previously plotted in that window, and are then stored in the pair of variates specified by the Y and X parameters.

Any number of points may be read in one DREAD statement. If the required number of points is known in advance, the Y and X variates can be declared with the appropriate length, and the input will terminate automatically when sufficient points have been read. Alternatively, if the lengths of the variates have not been defined in advance, points are read until you terminate the input, and the variates are defined accordingly. This action can be requested explicitly by setting option SETNVALUES=yes; the existing variate lengths are then ignored and points are read until the input is terminated. Graphical input can usually be terminated in two ways, either by pressing

a mouse button (usually the right button) or a key that has been specifically defined for this purpose, or by attempting to read a point lying outside the current axes. In case of difficulty you should refer to the *Users' Note* which will explain how to terminate the input on specific devices. The final point read as a terminator is not included in the `Y` and `X` variates. If you try to terminate input prematurely when a set number of values is to be read, the corresponding `Y` and `X` values are set to missing values.

The `PRINT` option of `DREAD` is similar to the `PRINT` option of `READ`. Putting `PRINT=data` lists the y- and x-values of the points that have been read, while `PRINT=summary` generates the usual summary of mean, minimum, maximum and number of values.

Several types of cursor may be available; again this will depend on the graphics device. The cursor is selected by setting the `CURSORTYPE` option to an integer between 1 and 10. Normally cursors 1, 2 and 3 are different graphics cursors; for example, large cross-hair, arrow and small cross. Cursors 4 and 5 may be set up to provide special functions called *rubber-band* and *rubber-rectangle*.

A rubber-band cursor works by reading one point in the normal way (as if `CURSORTYPE` was set to 1). This defines an anchoring point for a line whose other end is attached to the cursor. As you move the cursor, the line will change direction and contract or expand, but always linking the fixed point to the current cursor position: hence the term "rubber-band". When you read the next point this will become the anchor point for a new rubber-band segment which you use whilst locating a third point, and so on until the required number of points have been read.

The rubber-rectangle works in a similar way, with the first point being read with a normal cursor. This defines the fixed point and the cursor is now regarded as being attached to the diagonally opposite corner of a rectangle which will contract and expand as you move the cursor around the screen. Reading the second point terminates the input; with a rubber-rectangle cursor Genstat will always read exactly two values, ignoring the `SETNVALUES` option and any predefined length of `Y` and `X`.

The rubber-band and rubber-rectangle types of cursors may not be available on all devices, in which case setting `CURSORTYPE` to 4 or 5 will use one of the simpler cursors. However, setting `CURSORTYPE` to 5 will always read just two points, regarded as being diagonally opposite corners of a rectangle, whether or not the rubber-rectangle appears on the screen.

Some devices may have more than one method of manipulating the graphics cursor, for example by use of a joystick or mouse. In this case, cursor-types 1 to 5 will be set up as described above for the joystick, say, and types 6 to 10 will be the same types of cursor but controlled by the mouse. Usually, however, there will be only one method of control, in which case cursor-types 6 to 10 will be the same as types 1 to 5.

The `PEN` parameter of `DREAD` can be used to specify a pen which will be used to plot each point as its position is read. The various attributes of this pen determine how the points are plotted; these can be modified, in the usual way, using the `PEN` directive. If the pen method is set to `line`, `monotonic`, `open` or `closed`, then straight line segments will be drawn between the points; otherwise just the points themselves are plotted. If the points are to be joined by lines and a rubber-rectangle cursor is being used, the rectangle will be drawn rather than the diagonal line. If labels are set for the pen, they will be used in turn to mark the points as they are read; if the number of points exceeds the number of labels the labels will be recycled.

The `YGIVEN` and `XGIVEN` parameters allow you to identify points that have been plotted in an existing graph. They should be set to the y- and x-variates that were plotted on the graph. Each point that is read by `DREAD` is then located within this pair of variates, by finding the original point that is physically nearest to the new point, ignoring any differences in the scales of the y- and x-values. The unit number of the located points can be saved in a variate specified by the `SAVESET` parameter, and their coordinates in a pair of variates supplied by the `YSAVE` and `XSAVE` parameters. The length of the variates is defined in the same way as for the `Y` and `X` variates. The variates saved by `YSAVE` and `XSAVE` contain the actual coordinates of the plotted points that were

selected by DREAD; whereas the Y and X variates contain the coordinates of the exact position of the cursor. The SAVESET variate indicates the unit numbers of the selected points. This information could be used, for example, in CALCULATE or RESTRICT statements to refer to the units that have been identified on the graph. For example,

```
DREAD U; V; YGIVEN=Y; XGIVEN=X; SAVESET=SS
RESTRICT Y,X; .NOT.EXPAND(SS; NVALUES(X))
```

would have the effect of excluding the points identified by DREAD; in this example the exact cursor locations recorded in U and V are not of interest.

When the PEN parameter is being used to mark the points that are read, you may want to pause at the end of the read so that you can inspect the modified graph. This is controlled by the ENDACTION parameter.

Options: PRINT, CHANNEL, WINDOW, CURSORTYPE, SETNVALUES, ENDACTION.
Parameters: Y, X, YGIVEN, XGIVEN, SAVESET, PEN, YSAVE, XSAVE.

See also
Directive: DEVICE.
Genstat Reference Manual 1 Summary section on: Graphics.

DROP

Drops terms from a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode, summ, esti
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate, unchanged); default unch
CONSTANT = *string token*	How to treat the constant (estimate, omit, unchanged, ignore); default unch
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
AOVDESCRIPTION = *text*	Description for line in accumulated analysis of variance (or deviance) table when POOL=yes

Parameter

formula	List of explanatory variates and factors, or model formula

Description

DROP deletes terms from the current regression model, which may be linear, generalized linear, generalized additive, standard curve or nonlinear. It is best to give a TERMS statement before investigating sequences of models using DROP, in order to define a common set of units for the models that are to be explored. If no model has been fitted since the TERMS statement, the current model is taken to be the null model.

The model fitted by DROP will include a constant term if the previous model included one, and

will not include one if the previous model did not. You can, however, change this using the CONSTANT option.

The options of DROP are the same as those of the FIT directive, but with the extra NONLINEAR option which is relevant when fitting curves. For example, if we have a variate Dilution and a factor Solution, the program below will fit curves with separate linear and nonlinear parameters for the different solutions.

```
MODEL Density
TERMS Dilution * Solution
FITCURVE [PRINT=model,estimates; CURVE=logistic; \
  NONLINEAR=separate] Dilution * Solution
```

If we then put

```
DROP [NONLINEAR=common]
```

the curves will be constrained to have common nonlinear parameters, but all linear parameters will still be estimated separately for each group.

Options: PRINT, NONLINEAR, CONSTANT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION, PROBABILITY, AOVDESCRIPTION.
Parameter: unnamed.

See also
Directives: MODEL, TERMS, FIT, FITCURVE, FITNONLINEAR, ADD, SWITCH, TRY.
Functions: COMPARISON, POL, REG, LOESS, SSPLINE.
Genstat Reference Manual 1 Summary section on: Regression analysis.

DSAVE

Saves the current graphics environment settings to an external file.

No options

Parameters

FILENAME = *text*	File in which to save the environment settings
DESCRIPTION = *text*	Description for these settings

Description

High-resolution graphics in Genstat takes place in a "graphics environment" that specifies exactly how the display is produced. So it controls aspects like whether or not boxes are drawn around the plots, the positioning of the plots on the graphics frame, the styles of axes, and the colours and symbols of points. There are commands to modify all of these aspects, so that you can customize your graphs as required for a particular situation:

FRAME	defines the positions of the plotting windows within the graphics frame (or screen)
XAXIS	defines the x-axis in a window
YAXIS	defines the y-axis in a window
ZAXIS	defines the z-axis in a window
PEN	defines properties of the graphics "pens"
COLOUR	defines the colour map

To simplify the future plotting of graphs in the same style, the DSAVE directive allows you to save the current settings of the graphics environment to an external file. You can then use the DLOAD directive to read them back into Genstat, so that you can produce similar plots in future.

The FILE parameter gives the name of the file in which to save the settings. You can also set the DESCRIPTION parameter to a text containing a one-line description of the settings. This could be used, for example, to note that they were designed for a particular type of publication or report.

Options: none.
Parameters: FILENAME, DESCRIPTION.

See also

Directives: DLOAD, FRAME, XAXIS, YAXIS, ZAXIS, PEN, DEVICE, COLOUR.
Procedures: DHELP, FFRAME, GETRGB.
Genstat Reference Manual 1 Summary section on: Graphics.

DSHADE

Plots a shade diagram of 3-dimensional data.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the graph; default 1
KEYWINDOW = *scalar*	Window number for the key (0 for no key); default 2
YORIENTATION = *string token*	Y-axis orientation of the plot (reverse, normal); default reve
GRIDMETHOD = *string token*	How to draw a grid around the elements of the matrix (present, complete); default pres
PENGRID = scalar	Pen to use for the grid; default −7
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *symmetric matrix*, *matrix*, *table* or *pointer to variates*	Data to be plotted
PEN = *scalar* or *variate*	How to draw each shade
LIMITS = *variate*	Boundary values for changes in shade
NGROUPS = *scalar*	Number of groups to form from the data values (i.e. number of different shades)
INTERVAL = *scalar*	Interval between changes in shade
DESCRIPTION = *text*	Annotation for key

Description

DSHADE produces a shaded representation of a rectangular or symmetric matrix using high-resolution graphics. Each element of the data matrix is represented by a shaded rectangle indicating the value at that location, using its colour. This type of display is often used in a cluster analysis to display a similarity matrix, but it is also useful for the graphical display of spatial data.

The data are specified by the GRID parameter, in either a matrix, a symmetric matrix (e.g. of similarities), a 2-way table or a pointer to a set of variates.

The range of data values corresponding to each shade are determined using the NGROUPS, the LIMITS or the INTERVAL parameter. The first possibility is to set LIMITS to a variate defining the boundaries on the data values where the shades change. Alternatively, if LIMITS is unset, NGROUPS can be used to define the required number of shades; Genstat then partitions the range of data values into that number of equal intervals (and shades each interval in a different way). Or, if both NGROUPS and LIMITS are unset, INTERVAL can set the interval between each change in shade. Finally, if none of these parameters is set, Genstat uses a different shade for each distinct data value. Missing values are ignored, thus leaving blank areas in the plot.

By default, the shades are drawn using pens 1, 2 onwards, with pen 1 being used for the lowest data values. Alternatively, you can specify the pen or pens explicitly, using the PEN parameter. If PEN is set to a scalar, the shades are defined in increasing intensities of the colour of the specified pen. Alternatively, if PEN is set to a variate of length two, the pens are taken to define the shades of the minimum and maximum data values, and the other shades are interpolated

between them. Finally, you can set PEN to a variate with more than two values, and the shades use the pens in the order in which they are given in the variate (recycling if insufficient pens are defined for the total number of shades).

The shades are controlled by the current COLOUR settings of the pens. If the default settings do not produce a suitable display, these attributes should be set by a PEN statement before using DSHADE.

The GRIDMETHOD option specifies whether an outline should be drawn around each element of the matrix. The default setting, present, produces an outline for all values that are present; i.e. it ignores missing values. This is suitable where data have been sampled over an irregularly shaped area. Alternatively, with the complete setting, an outline is drawn around every element. Setting GRIDMETHOD=* stops the grid being drawn, which may be preferable if there are a large number of elements in the input data. The PENGRID option specifies which pen to use to draw the grid. The default is to use pen -7.

The YORIENTATION option controls the orientation of the y-axis. By default this is reversed, so that the data are in the same order as they would take if the data matrix were printed.

The TITLE, WINDOW, SCREEN and ENDACTION options are used to specify a title, the plotting window, whether the screen should be cleared first, and whether there should be a pause once the plotting is finished; as in other graphics directives (see, for example, DGRAPH). Similarly, the KEYWINDOW and KEYDESCRIPTION options and the DESCRIPTION parameters allow a key to be defined, if feasible for these plots with the current graphics device.

Options: TITLE, WINDOW , KEYWINDOW , YORIENTATION, GRIDMETHOD, PENGRID, SCREEN, KEYDESCRIPTION, ENDACTION.
Parameters: GRID, PEN, LIMITS, NGROUPS, INTERVAL, DESCRIPTION.

Action with RESTRICT

DSHADE takes account of restrictions on any of the variates in a GRID pointer.

See also

Directives: DBITMAP, DCONTOUR, DSURFACE, D3HISTOGRAM, FRAME, XAXIS, YAXIS, PEN, MATRIX, POINTER, SYMMETRICMATRIX, TABLE.
Procedure: DXYDENSITY.
Genstat Reference Manual 1 Summary section on: Graphics.

DSTART

Starts a sequence of related high-resolution plots.

Options

TITLE = *text*	Overall title for the plots
PEN = *scalar*	Pen to use for the title; if this is not set, pen −12 is used

Description

This directive is relevant when a sequence of high-resolution graphics commands (such as DGRAPH and DHISTOGRAM) is to be used to build up a complicated plot. The information from each command is then accumulated by Genstat, but no high-resolution plots are drawn until a DFINISH command is received. The TITLE option can specify an overall title, and PEN can specify the pen to use. If PEN is not set, the title is plotted using pen −12.

Options: TITLE, PEN.
Parameters: none.

See also

Directives: DFINISH, DCLEAR.
Genstat Reference Manual 1 Summary section on: Graphics.

DSURFACE

Produces perspective views of a two-way arrays of numbers.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ELEVATION = *scalar*	The elevation of the viewpoint relative to the surface; default 25 (degrees)
AZIMUTH = *scalar*	Rotation about the horizontal plane; the default of 225 degrees ensures that, with a square matrix M, the element M$[1;1] is nearest to the viewpoint
DISTANCE = *scalar*	Distance of the viewpoint from the centre of the grid on the base plane; default * gives a distance of 100 times the maximum of the x-range and the y-range
ZSCALE = *scalar*	defines the scaling of the z-axis relative to the horizontal (x-y) axes; default 1
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *identifier*	Pointer (of variates representing the columns of a data matrix), matrix or two-way table specifying values on a rectangular grid
PEN = *scalar*	Pen number to be used for the plot; default 1
PENFILL = *scalar* or *variate*	Pen number(s) defining how to fill the areas between contours (0 or * leaves the areas in the background colour); default 3
PENMESH = *scalar*	Pen number to use to draw the mesh (omitted if set to 0 or *); default 1
PENSIDE = *scalar*	Pen number to use to shade the sides of the surface (omitted if set to 0 or *); default *
NCONTOURS = *scalar*	Number of contours; default 10
CONTOURS = *variate*	Positions of contours
INTERVAL = *scalar*	Interval between contours
DESCRIPTION = *text*	Annotation for key

Description

The DSURFACE directive produces a perspective (or *conical*) projection of a surface, showing the view from a particular viewpoint. The surface is represented by a grid of z-values or heights. The grid can be a rectangular matrix, a two-way table or a pointer to a set of variates; the y-dimension is represented by the rows of the structure and the x-dimension by the columns. In each case there must be at least three rows and three columns of data (after allowing for any restrictions on a set of variates). Missing values are not permitted; that is, only complete grids can be displayed. If the grid is supplied as a table with margins, these will be ignored when plotting the surface.

The position of the viewpoint is specified in polar coordinates, using the options ELEVATION, DISTANCE and AZIMUTH. These define the angle of elevation, in degrees, above the base plane of the surface, distance from the centre of this plane, and angular position relative to the vertical z-axis, respectively. The default settings of ELEVATION, DISTANCE and AZIMUTH have been chosen to produce a reasonable display of most surfaces; but if, for example, some parts of the surface are obscured by high points they can be modified to obtain a better view. Altering the value of AZIMUTH will, in effect, rotate the surface in the horizontal plane about a vertical axis drawn through the centre of the grid; the default value of 225 degrees ensures that the element in the first row and column of the grid is at the corner nearest the viewpoint. Small values of DISTANCE produce a perspective view; larger values, like the default of 100 times the maximum of the x-range and the y-range, effectively put the viewpoint at infinity to produce an "orthographic parallel projection".

The ZSCALE option specifies a scaling factor for the z-axis (or vertical axis) of the plotted surface. Generally values between 0.5 and 2.0 are most successful; large values result in a flatter surface, while smaller values produce a steep surface, accentuating changes in the data.

The TITLE, WINDOW, SCREEN and ENDACTION options are used to specify a title, the plotting window, whether the screen should be cleared first, and whether there should be a pause once the plotting is finished; as in other graphics directives (see, for example, DGRAPH). Similarly, the KEYWINDOW and KEYDESCRIPTION options and the DESCRIPTION parameters allow a key to be defined, if feasible for these plots with the current graphics device.

The PEN parameter specifies the pen to be used to plot the surface (by default, pen 1). The PEN directive can be used to modify the colour and the thickness of the pen, but the other attributes of the pen are ignored.

The NCONTOURS, CONTOURS and INTERVAL parameters control the contours drawn on the surface, if these are available on the current graphics device. The first possibility is to define the contours explicitly using the CONTOURS parameter. Alternatively, if CONTOURS is unset, INTERVAL can set the required interval between each contour. Or, if both CONTOURS and INTERVAL are unset, NCONTOURS defines the required number of lines. Genstat then partitions the range of data values accordingly to give NCONTOURS evenly-spaced contours (or fewer contours if there are insufficient distinct grid values).

The PENFILL parameter defines how to shade the areas between the contours. If this is set to a scalar, the shades are defined in increasing intensities of the colour of the specified pen. Alternatively, if PENFILL is set to a variate of length two, the pens are taken to define the shades at the minimum and maximum heights, and the other shades are interpolated between them. Finally, you can set PENFILL to a variate with more than two values, and the shading uses the pens in the order in which they are given in the variate (recycling if insufficient pens are defined for the total number of contours). The default is to use pen 3. However, if you set PENFILL to 0 or to a missing value, there will be no shading (that is, the areas between the contours will be in the background colour).

The PENMESH parameter specifies a pen to be used to draw a mesh on the surface. This consists of lines marking the points of the surface that lie above a rectangular grid on the xy plane. By default pen 1 is used, but if you set PENMESH to 0 or to a missing value the mesh is omitted.

The PENSIDE parameter defines the pen to use to shade the sides of the surface. There is no shading if this is set to 0 or a missing value, which is the default. The CFILL setting of the pen (see the PEN directive) specifies which colour is used.

Simple axes are drawn to indicate the directions in which x and y increase. The TITLE parameter of the XAXIS and YAXIS directives can be used to add further annotation.

Options: TITLE, WINDOW, KEYWINDOW, ELEVATION, AZIMUTH, DISTANCE, ZSCALE, SCREEN, KEYDESCRIPTION, ENDACTION.

Parameters: GRID, PEN, PENFILL, PENMESH, PENSIDE, NCONTOURS, CONTOURS, INTERVAL, DESCRIPTION.

Action with RESTRICT

DSURFACE takes account of restrictions on any of the variates in a GRID pointer.

See also

Directives: DBITMAP, DCONTOUR, DSHADE, D3HISTOGRAM, FRAME, XAXIS, YAXIS, ZAXIS, PEN, MATRIX, POINTER, TABLE.

Genstat Reference Manual 1 Summary section on: Graphics.

DUMMY

Declares one or more dummy data structures.

Options

VALUE = *identifier*	Value for all the dummies; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the dummies in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the dummies
VALUE = *identifiers*	Value for each dummy
EXTRA = *texts*	Extra text associated with each identifier

Description

The IDENTIFIER parameter lists the identifiers of the dummies that are to be declared. Dummies store the identifiers of other structures. These are particularly useful when you want the same series of statements to be used with several different data structures. By using a dummy structure within the statements, you can make them apply to whichever structure you require. The dummy structure is like a plug which can be connected to the structure that you need; the important point is that you can then connect another structure without changing the statements themselves. In nearly all identifier lists Genstat will replace a dummy by the identifier that it stores. The only exceptions are the IDENTIFIER parameter of the DUMMY directive itself, the STRUCTURE parameter of ASSIGN, the parameters of FOR, and in the UNSET function in expressions. (The most obvious occasions where this is useful are in loops and procedures, and there the dummies are declared automatically.)

Values can be assigned to the dummies by either the VALUE option or the VALUE parameter. The option defines a common value for all the structures in the declaration, while the parameter allows the structures each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

You can associate a text of extra annotation with each dummy using the EXTRA parameter.

If MODIFY is set to yes any existing attributes and values of the dummies are retained; otherwise these are lost. The IPRINT option can be set to specify how the dummies will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned.

Options: VALUE, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUE, EXTRA.

See also

Directives: ASSIGN, FOR, PROCEDURE.
Genstat Reference Manual 1 Summary section on: Data structures.

DUMP

Prints information about data structures, and internal system information.

Options

PRINT = *string tokens*	What information to print about structures (attributes, values, identifiers, space); default attr
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
INFORMATION = *string tokens*	What information to print for each structure (brief, full, extended); default brie
TYPE = *string tokens*	Which types of structure to include in addition to those in the parameter list (all, ASAVE, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, RSAVE, scalar, SSPM, symmetricmatrix, table, text, tree, TSAVE, TSM, variate, VSAVE); default * i.e. none
SYSTEM = *string token*	Whether to display Genstat system structures (yes, no); default no
UNNAMED = *string token*	Whether to display unnamed structures (yes, no); default no

Parameter

identifiers or *numbers*	Identifier or reference number of a structure whose information is to be printed

Description

The structures for which the information is to be displayed are specified by the parameter of DUMP. The PRINT option indicates what is to be presented: you can ask for just the identifiers, or values and identifiers, or attributes (the identifier is itself an attribute), or for all three. For example, to get all three for the structures A and B you would put:

```
DUMP [PRINT=attributes,values] A,B
```

There is also a setting, space, which provides information about the current use of workspace within Genstat.

If the CHANNEL option is set to a scalar, this specifies the output channel to which the information is sent. Alternatively, if you specify the identifier of a text structure, the lines of information will be stored in the text instead of being printed; likewise if you specify the identifier of a structure that has not yet been declared, it will be defined automatically as a text to store the information. If CHANNEL is not specified, the information is displayed on the current output channel.

The INFORMATION option selects which attributes are presented. The default setting brief selects only the most important ones. The setting full causes all the attributes to be presented, and the setting extended also gives details of the structures associated with listed structures.

Some of the attributes may be set to unnamed structures. You can obtain further information about these by setting option UNNAMED=yes. Alternatively, you can dump a specific unnamed structure by giving its (negative) reference number (as displayed by DUMP when indicating its association with another structure) in the parameter list. This is likely to be useful mainly to advanced users.

The TYPE option lets you display, in addition, lists of all structures of a particular type, or of several types. For example, if you had forgotten the identifier of a factor, you could give the statement

```
DUMP [TYPE=factor; PRINT=identifiers]
```

This lists all the current factors. When `PRINT=attributes` or `values` (or both), the setting `TYPE=all` provides a list of all named and unnamed structures, except system structures. Setting `PRINT=identifiers` with `TYPE=all` lists only named structures.

The `SYSTEM` option allows all the system structures to be dumped: there are many of these, so it is not a good idea to set this option frivolously.

Options: `PRINT`, `CHANNEL`, `INFORMATION`, `TYPE`, `SYSTEM`, `UNNAMED`.
Parameter: unnamed.

See also
Directives: `GETATTRIBUTE`, `LIST`.
Genstat Reference Manual 1 Summary section on: Data structures.

DUPLICATE

Forms new data structures with attributes taken from an existing structure.

Options

ATTRIBUTES = *string tokens*	Which attributes to duplicate (all, nvalues, values, nlevels, levels, labels (of factors or pointers), extra, decimals, characters, rows, columns, classification, margins, suffixes, minimum, maximum, restriction, referencelevel); default all
REDEFINE = *string token*	Whether or not to delete the attributes of the new structures beforehand so that their types can be redefined (yes, no); default no

Parameters

OLDSTRUCTURE = *identifiers*	Data structures to provide attributes for the new structures
NEWSTRUCTURE = *identifiers*	Identifiers of the new structures
VALUES = *identifiers*	Values for each new structure
DECIMALS = *scalars*	Number of decimals for printing numerical structures
CHARACTERS = *scalars*	Number of characters for printing texts or labels of a factor
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for numerical structures
MAXIMUM = *scalars*	Maximum value for numerical structures

Description

The DUPLICATE directive allows you to define new data structures with attributes like those of existing structures. The attributes to be duplicated are defined by the ATTRIBUTES option. The structures from which the attributes are to be taken are specified by the OLDSTRUCTURES parameter, while the structures that are to be defined are specified by the NEWSTRUCTURES parameter. The other parameters allow some of the more important attributes to be reset at the same time. For example, here the factor Species2 takes its levels (and thus its number of levels) from the factor Species1. However, the labels are not transferred, and other values are defined using the VALUES parameter.

```
FACTOR [LEVELS=!(0,1); LABELS=!T(absent,present); \
  VALUES=0,1,1,0,0,0,1] Species1
DUPLICATE [ATTRIBUTES=levels] Species1; \
  NEWSTRUCTURE=Species2; VALUES=!(1,0,1,1,0,1,0)
```

You can set option REDEFINE=yes, to allow DUPLICATE to change the type of any pre-defined new structure, if necessary, to have the same type as the corresponding old structure. Otherwise, DUPLICATE will report a fault if the new structure has previously been defined to have a different type.

Options: ATTRIBUTES, REDEFINE.

Parameters: OLDSTRUCTURE, NEWSTRUCTURE, VALUES, DECIMALS, CHARACTERS, EXTRA, MINIMUM, MAXIMUM.

See also
Directive: RENAME.
Procedure: PDUPLICATE.
Genstat Reference Manual 1 Summary section on: Data structures.

D3GRAPH

Plots a 3-dimensional graph.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ELEVATION = *scalar*	The elevation of the viewpoint relative to the surface; default 25 (degrees)
AZIMUTH = *scalar*	Rotation about the horizontal plane; the default of 225 degrees ensures that a point at the minimum x- and y-value is nearest to the viewpoint
DISTANCE = *scalar*	Distance of the viewpoint from the centre of the grid on the base plane; default * ensures that the data points fill the viewing area
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep, resize); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

X = *identifiers*	X-coordinates
Y = *identifiers*	Y-coordinates
Z = *identifiers*	Z-coordinates
PEN = *scalars* or *variates* or *factors*	Pen number for each graph (use of a variate or factor allows different pens to be defined for different sets of units); default * uses pens 1, 2, and so on for the successive graphs
DESCRIPTION = *texts*	Annotation for key
UNITNUMBERS = *identifiers*	Specifies unit numbers to be used when points are selected in the graphics viewer; default * uses the actual unit numbers of the values in the X and Y structures

Description

The D3GRAPH directive produces high-resolution graphs, containing points, lines or filled shapes in three dimensions. The graph is produced on the current graphics device which can be selected using the DEVICE directive. The WINDOW option defines the window, within the plotting area, in which the graph is drawn; by default this is window 1.

The position of the viewpoint is specified in polar coordinates, using the options ELEVATION, DISTANCE and AZIMUTH. These define the angle of elevation, in degrees, above the base plane of the surface, distance from the centre of this plane, and angular position relative to the vertical z-axis, respectively.

The default settings of ELEVATION, DISTANCE and AZIMUTH have been chosen to produce a reasonable display of most situations; but if, for example, some parts of the plot are obscured they can be modified to obtain a better view. Altering the value of AZIMUTH will, in effect, rotate the plot in the horizontal plane about a vertical axis drawn through the centre of the plot; the default value of 225 degrees ensures that a point with the minimum x- and y-value would be at the corner nearest the viewpoint.

The X, Y and Z parameters specify the coordinates of the points to be plotted; they must be numerical structures (scalars, variates, factors, matrices or tables) of equal length. If any of the variates or factors is restricted, only the subset of values specified by the restriction will be included in the graph. The restrictions are applied to the X, Y and Z variates or factors in parallel sets, and do not carry over to all the variates or factors in a list. Any associated structures, like variates specified by the PEN parameter or factors used to provide labels for the points, must be of the same length as X, Y and Z.

Each set of X, Y and Z structures has an associated pen, specified by the PEN parameter. By default, pen 1 is used for the first set, pen 2 for the second, and so on. The type of graph that is produced is determined by the METHOD setting of that pen. This can be point, to produce a point plot or scatterplot; line to join the points with straight lines; or fill to produce shaded objects. In the initial graphics environment, all the pens are defined to produce point plots. This can be modified using the METHOD option of the PEN directive. Other attributes of the pen can be used to control the colour, font, symbols and labels.

With METHOD=fill, the points defined by the X, Y and Z variates are joined by straight lines to form one or more polygons or polyhedrons which are then filled in the colour specified for the pen. The JOIN parameter of PEN is ignored for this directive. The points are plotted in the order in which they occur in the data.

A warning message is printed if the data contain missing values. The effect of these depends on the type of graph being produced, as follows. If the method is point there will be no indication on the graph itself that any points were missing (but obviously none of the points with missing values for either the x-, y- or z-coordinate can be included in the plot). If a line is plotted through the points there will be a break wherever a missing value is found; that is, line segments will be omitted between points that are separated by missing values. When using METHOD=fill missing values will, in effect, define subsets of points, each of which will be shaded separately.

The PEN parameter can also be set to a variate or factor, to allow different pens to be used for different subsets of the units. With a factor, the units with each level are plotted separately, using the pen defined by the level concerned. If PEN is set to a variate, its values similarly define the pen for each unit. For example, if you fit separate regression lines to some grouped data, you can easily plot the fitted lines in just two statements, one to set up the pens and one to plot the data:

```
PEN 1...Ngroups; METHOD=line; SYMBOL=0
D3GRAPH Fitted; X1; X2; PEN=Groups
```

By default, Genstat calculates bounds on the axes that are wide enough to include all the data; the range of the data is extended by five percent at each end, and the axes are drawn on the left-hand side and bottom edge of the graph. This can all be changed by the XAXIS, YAXIS and ZAXIS directives using the LOWER and UPPER parameters to set the bounds, and XORIGIN, YORIGIN and ZORIGIN to control the positions of the axes. Other parameters allow you to control the axis labelling and style. If the axis bounds are too narrow, some points may be excluded from the graph, so that *clipping* occurs. If the plotting method is point, Genstat ignores points that are out of bounds. For other settings of METHOD, lines are drawn from points that are within bounds towards points that are out of bounds, terminating at the appropriate edge. Clipping may also occur if the method is monotonic, open or closed and you have left Genstat to set default axis bounds, because these methods fit curves that may extend beyond the boundaries. If this occurs you should use the relevant axis directive to provide increased axis bounds. When you use several D3GRAPH statements with SCREEN=keep to build up a complex graph, the axes are drawn only the first time, and the same axes bounds are then used for the subsequent graphs. You should then define axis limits that enclose all the subsequent data. Alternatively, if you set SCREEN=resize, the axes and their bounds will be adjusted, if necessary, to enclose the additional information. Axes are drawn only if SCREEN=clear, or the specified window has not been used since the screen was last cleared, or the window has been redefined by a FRAME statement.

The KEYWINDOW option specifies the window in which the key appears; by default this is window 2. Alternatively, you can set KEYWINDOW=0 to suppress the key. The key contains a line of information for each pair of Y and X structures, written with the associated pen. This will indicate the symbol used, the line style (for a plotting method of line or curve) or a shaded block to illustrate the colour (when METHOD=fill), the name of the structure (if any) defined by the LABELS parameter of PEN, and a description indicating the identifiers of the data plotted (for example Residuals v Fitted). Alternatively, you can supply your own key, using the DESCRIPTION parameter, and you can specify a title for the key using the KEYDESCRIPTION option. If you draw several graphs using SCREEN=keep or SCREEN=resize and the same key window, each new set of information is appended to the existing key, until the window is full.

If you have set the PEN parameter to a variate or factor in order to plot independent subsets of the data, the key will contain information for each subset. If the LABELS parameter of PEN has been used to specify labels for the points, each line of the key will contain the label corresponding to the first value of the subset, rather than the identifier of the labels structure itself.

The Graphics Viewer has a tool that allows you to select points, and copy their unit numbers onto the clipboard. Usually these numbers are simply the locations of the plotted values in the X and Y structures. However, you can use the UNITNUMBERS parameter to supply other numbers. (This may be useful if, for example, you are plotting sorted values.)

The TITLE option can be used to provide a title for the graph. You can also put titles on the axes by using the TITLE parameters of the XAXIS, YAXIS and ZAXIS directives. The SCREEN option controls whether the graphical display is cleared before the graph is plotted, and the ENDACTION option controls whether Genstat pauses at the end of the plot.

Options: TITLE, WINDOW, KEYWINDOW, ELEVATION, AZIMUTH, DISTANCE, SCREEN, KEYDESCRIPTION, ENDACTION.
Parameters: X, Y, Z, PEN, DESCRIPTION, UNITNUMBERS.

Action with RESTRICT

You can arrange to plot only a subset of the points specified by a particular set of X, Y and Z vectors and associated PEN vector, by restricting any one of them. If more than one of these is restricted, then they must all be restricted in exactly the same way.

See also

Directives: DGRAPH, D3HISTOGRAM, FRAME, XAXIS, YAXIS, ZAXIS, PEN.
Procedures: DSCATTER, DXYDENSITY, TRELLIS.
Genstat Reference Manual 1 Summary section on: Graphics.

D3HISTOGRAM

Plots three-dimensional histograms.

Options

TITLE = *text*	General title; default *
WINDOW = *scalar*	Window number for the plots; default 1
KEYWINDOW = *scalar*	Window number for the key (zero for no key); default 2
ELEVATION = *scalar*	The elevation of the viewpoint relative to the surface; default 25 (degrees)
AZIMUTH = *scalar*	Rotation about the horizontal plane; the default of 225 degrees ensures that, with a square matrix M, the element M$[1;1] is nearest to the viewpoint
DISTANCE = *scalar*	Distance of the viewpoint from the centre of the grid on the base plane; default * gives a distance of 100 times the maximum of the x-range and the y-range
SCREEN = *string token*	Whether to clear the screen before plotting or to continue plotting on the old screen (clear, keep); default clea
KEYDESCRIPTION = *text*	Overall description for the key; default *
ENDACTION = *string token*	Action to be taken after completing the plot (continue, pause); default * uses the setting from the last DEVICE statement

Parameters

GRID = *identifier*	Pointer (of variates representing the columns of a data matrix), matrix or two-way table specifying values on a regular grid
PEN = *scalar*	Pen number to be used for the plot; default 3
DESCRIPTION = *texts*	Annotation for key

Description

D3HISTOGRAM plots a surface as a three-dimensional (or bivariate) histogram. The surface is represented by a grid of z-values or heights. The grid can be a rectangular matrix, a two-way table or a pointer to a set of variates; the y-dimension is represented by the rows of the structure and the x-dimension by the columns. In each case there must be at least three rows and three columns of data (after allowing for any restrictions on a set of variates). Missing values are not permitted; that is, only complete grids can be displayed. If the grid is supplied as a table with margins, these will be ignored when plotting the surface.

The position of the point from which the histogram is viewed is specified in polar coordinates, using the options ELEVATION, DISTANCE and AZIMUTH. These define the angle of elevation, in degrees, above the base plane of the surface, distance from the centre of this plane, and angular position relative to the vertical z-axis, respectively, similarly to the DSURFACE directive.

The TITLE, WINDOW, SCREEN and ENDACTION options are used to specify a title, the plotting window, whether the screen should be cleared first, and whether there should be a pause once the plotting is finished; as in other graphics directives (see, for example, DGRAPH). Similarly, the KEYWINDOW and KEYDESCRIPTION options and the DESCRIPTION parameters allow a key to be defined, if feasible for these plots with the current graphics device.

The PEN parameter specifies the pen to be used to plot the histogram (by default, pen 3). The PEN directive can be used to modify the colour and the thickness of the pen, but the other attributes of the pen are ignored.

Simple axes are drawn to indicate the directions in which x and y increase. The TITLE

parameter of the XAXIS and YAXIS directives can be used to add further annotation.

Options: TITLE, WINDOW, KEYWINDOW, ELEVATION, AZIMUTH, DISTANCE, SCREEN, KEYDESCRIPTION, ENDACTION.
Parameters: GRID, PEN, DESCRIPTION.

Action with RESTRICT
D3HISTOGRAM takes account of restrictions on any of the variates in a GRID pointer.

See also
Directives: DBITMAP, DCONTOUR, DSHADE, DSURFACE, D3GRAPH, BARCHART, DHISTOGRAM, DPIE, LPHISTOGRAM, FRAME, XAXIS, YAXIS, ZAXIS, PEN, MATRIX, POINTER, TABLE.
Procedures: TRELLIS, DBARCHART, DOTHISTOGRAM, DOTPLOT, DCIRCULAR, WINDROSE.
Genstat Reference Manual 1 Summary section on: Graphics.

EDIT

Edits text vectors.

Options

CHANNEL = *scalar* or *text*	Text structure containing editor commands or a scalar giving the number of a channel from which they are to be read; default is the current input channel
END = *text*	Character(s) to indicate the end of the commands read from an input channel; default is the character colon (:)
WIDTH = *scalar*	Limit on the line width of the text; default *
SAVE = *text*	Text to save the editor commands for future use; default *

Options

Parameters

OLDTEXT = *texts*	Texts to be edited
NEWTEXT = *texts*	Text to store each edited text; if any of these is omitted, the corresponding OLDTEXT is used

Description

The EDIT directive edits each text in the OLDTEXT list, storing the results in the corresponding structure in the NEWTEXT list. It both edits and stores each text before moving on to the next. If you have not already declared any of the texts in the NEWTEXT list, it will be declared implicitly. If you give a missing identifier (*) in the NEWTEXT list, the edited version simply replaces the values of the original: thus the old text will be overwritten by the new text. You can also omit a text from the OLDTEXT list; you might do this if you wanted to form the values of the new text entirely from within the editor.

The CHANNEL option tells Genstat where to find the editing commands. A scalar specifies the number of an input channel from which the commands are to be read. Alternatively, you can specify a text structure containing the commands. In either case the commands should be terminated by the string specified by the END option. The end string can be more that one character; the default is the single character colon (:). Genstat gives a warning if you have forgotten to specify the end string in a text of commands. The default for the CHANNEL option is to take input from the current input channel.

The WIDTH option specifies the maximum line length for vectors of commands and of text, the default being 80 and the maximum being 255.

The SAVE option allows you to specify a text structure to store the edit commands, so that you can save them for future EDIT statements.

Commands for the editor can be given in either upper or lower case. You can put as many commands as you like on a line, subject only to the width restriction set by the WIDTH option. Commands must be separated by at least one space. You cannot put spaces into the middle of a command, unless they are part of a character string (or part of a sequence of commands).

The character that separates the parts of a command is written here as /, but you can use any character for this other than a space or a digit.

Genstat puts the lines from the old text into an internal *buffer*, where they are modified according to the commands that you specify. While you are editing, Genstat moves a notional *marker* around the buffer. The marker can be moved backwards or forwards along a line or between lines. So you can move around the text and modify the lines in any order. Some commands move the marker automatically, as explained in the definitions below. If the marker is before the first line of text it is at the [start] position; if it is after the last line of text it is at the [end] position. The line that currently contains the marker is called the *current line*.

Genstat does not write anything to the new text until the edit has been completed (so if you use the `Q` command, the new text is left unaltered).

Some commands allow you to specify a number: for example `Dn` deletes the next *n* lines. Genstat gives a warning message if this number is zero or is not an integer.

The commands are as follows.

`A` Insert the next line of text from the buffer, immediately after the marker within the current line.

`B` Break the current line at the marker position. Text before the marker is written as a new line to the internal buffer and text after the marker becomes the new current line with the marker at character position 1.

`C` Cancel edits performed on the current line by restoring it to the form in which it was most recently read from the buffer. Note that if you have previously edited the line and then moved to some other line, it is the previously edited form that will be given, not the form as originally in the old text; also, if you have given any `A` or `B` commands during your modification of the current line, their effects are not negated, so for example any lines that have been inserted into the current line by `A` will be lost.

`D` Delete the current line, and make the next line the current line with the marker at character position 1.

`Dn` Delete the next *n* lines (including the current line), making the next line after that the current line with the marker at position 1.

`D+n` Synonymous with `Dn`.

`D+` is a synonym for `D` or `D+1`.

`D+*` Delete from the current line to end of text. The current line is then `[end]`.

`D*` Synonymous with `D+*`.

`D-` Delete the current line, making the previous line the current line with the marker at character position 1.

`D-n` Delete the current and previous *n* lines, making the line before that the current line with the marker at character position 1.

`D-` is a synonym for `D-1`.

`D-*` Delete the current line and all previous lines, the current line is then `[start]`.

`D/s/` Delete from the current line to the line with the next occurrence of the character string *s*. The marker is placed immediately before the character string *s* in the located line. If *s* occurs after the marker on the current line, the marker is moved up to *s* and no lines are deleted.

`D-/s/` The same as `D/s/`, except that it moves backwards through the text, deleting all lines from and including the current one until the first occurrence of a line containing the character string *s*. The marker is placed immediately before the located character string *s*. If *s* occurs before the marker on the current line, the marker placed before *s* and no lines are deleted.

`F/i/` Inserts the contents of the text structure with identifier *i* immediately before the current line. The marker is not moved.

`G+/s/t/` substitutes string *t* for all occurrences of string *s* found after the marker on the current and subsequent lines, and moves the marker to the end of the text.

`G/s/t/` is a synonym for `G+/s/t/`.

`G-/s/t/` substitutes string *t* for all occurrences of string *s* found before the marker on the current and previous lines, and moves the marker to the start of the text.

`I/s/` Inserts the string *s* as a new line immediately before the current line. The marker is not moved.

`L` Moves the marker to the start of the next line, which can be `[end]`.

`Ln` Moves the marker to the start of the *n*th line after the current line. So `L1` gives the next line.

`L+`*n* Is synonymous with `L`*n*.

`L+` Is synonymous with `L` or `L+1`.

`L+*` Moves the marker to `[end]`.

`L*` is a synonym for `L+*`.

`L-`*n* Moves the marker to the start of the *n*th line before the current line, which can be `[start]`. `L-1` gives the line immediately before the current line.

`L-` Is synonymous with `L-1`.

`L-*` Moves the marker to `[start]`.

`L+/`*s*`/` Moves the marker to the position immediately before the next occurrence of the character string *s* after the current marker position; this occurrence need not be on the current line. If the string *s* is not found, the marker will be located at `[end]`.

`L-/`*s*`/` Moves the marker to the position immediately before the first occurrence of the string *s* before the current marker position; this occurrence need not be on the current line. If the string *s* is not found, the marker will be located at `[start]`.

`P` moves the marker one character to the right along the current line.

`P+`*n* Moves the marker *n* characters to the right of the current position within the current line. You cannot move the marker beyond the maximum line length (which will vary between computers, but is normally the same as the width of your local line-printer).

`P+` is a synonym for `P` or `P+1`.

`P+*` Moves the marker to the position immediately after the last non-blank character in the current line. This can be to the left of the current marker position.

`P-`*n* Moves the marker *n* characters to the left of the current position within the current line. The marker cannot be moved to the left of character position 1.

`P-` is a synonym for `P-1`.

`P-*` Moves the marker to the position immediately before the first non-blank character after character position 1. This can be to the right of the current marker position.

`P`*n* Moves the marker to the character position *n* within the current line, counting from the left and starting at 1. The maximum value of *n* varies between computers but is normally the same as the width of your local line-printer.

`Q` Abandons the current edit, leaving the original text unaltered.

`R+/`*s*`/`*t*`/` substitutes character string *t* for the next occurrence of character string *s* after the marker on the current or subsequent lines, and moves the marker to the position immediately after *t*.

`R/`*s*`/`*t*`/` is a synonym for `R+/`*s*`/`*t*`/`.

`R-/`*s*`/`*t*`/` substitutes string *t* for the nearest occurrence of string *s* before the marker on the current or previous lines; the marker moves to be immediately before string *t*.

`S/`*s*`/`*t*`/` Substitutes the string *t* for the next occurrence of string *s* after the marker within the current line. The marker is moved to the character position immediately after the last character in *t*. If *s* is null (when the command is `S//`*t*`/`) then *t* is inserted immediately after the marker. If *t* is null (when the command is `S/`*s*`//`), then *s* is deleted from the line.

`V` Turns on the verification mode. Then, if you are working interactively, the current line will be displayed each time that Genstat prompts you for commands. By default the marker is indicated by the character > but you can change this by the command `V`*c* or `V+`*c*.

`V`*c* Turns on the verification mode (see `V`), and changes the marker character to *c*.

`V+`*c* Is synonymous with `V`*c*.

`V-` Turns verification mode off (see `V`).

`(`*cseq*`)`*n* Repeats the command sequence, *cseq*, *n* times. The command sequence *cseq* can be any valid combination of editing commands, each separated by at least one space. The complete sequence, including brackets and repeat count, must all be on a single line. You can nest sequences up to a depth of 10.

(*cseq*)* Repeats the command sequence cseq until [end] or [start] is encountered. In all other respects (*cseq*)* behaves exactly as (*cseq*)*n*; so it would be equivalent to putting *n* equal to some very large number.

Options: CHANNEL, END, WIDTH, SAVE.
Parameters: OLDTEXT, NEWTEXT.

Action with RESTRICT
If any of the old texts are restricted, they must all be restricted to exactly the same set of units. Then only those units will be involved in the edit. When a restriction is in force, you cannot add or delete any units (or lines).

See also
Directives: TEXT, CONCATENATE, EQUATE, TXBREAK, TXCONSTRUCT, TXFIND, TXPOSITION, TXREPLACE.
Procedure: APPEND, SUBSET, STACK, UNSTACK.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

ELSE

Introduces the default set of statements in block-if or in multiple-selection control structures.

No options or parameters

Description

The use of ELSE in *block-if* structures is explained in the description of the IF directive. Its use in *multiple-selection* control structures is explained in the description of CASE.

Options: none.
Parameters: none.

See also

Directives: CASE, OR, ENDCASE, IF, ELSIF, ENDIF, EXIT.
Genstat Reference Manual 1 Summary section on: Program control.

ELSIF

Introduces a set of alternative statements in a block-if control structure.

No options

Parameter

expression	Logical expression to indicate whether or not the set of statements is to be executed.

Description

A *block-if* structure consists of one or more alternative sets of statements. The first of these is introduced by an `IF` statement. There may then be further sets introduced by `ELSIF` statements. Then you can have a final set introduced by an `ELSE` statement, and the whole structure is terminated by an `ENDIF` statement. Full details are given in the description of the `IF` directive.

Options: none.
Parameter: unnamed.

See also

Directives: `IF`, `ELSE`, `EXIT`, `ENDIF`, `CALCULATE`.
Genstat Reference Manual 1 Summary section on: Program control.

ENDBREAK

Returns to the original channel or control structure and continues execution.

No options or parameters

Description

ENDBREAK ends breaks instigated by the BREAK directive, where more details are given.

Options: none.
Parameters: none.

See also

Directives: BREAK, DEBUG.
Genstat Reference Manual 1 Summary section on: Program control.

ENDCASE
Indicates the end of a "multiple-selection" control structure.

No options or parameters

Description
A *multiple-selection* control structure consists of several alternative blocks of statements. The first of these is introduced by a `CASE` statement. This has a single parameter, which is an expression that must yield a single number. Subsequent blocks are each introduced by an `OR` statement. There can then be a final block, introduced by an `ELSE` statement, as in the block-if structure. The whole structure is terminated by an `ENDCASE` statement. Full details are given in the description of the `CASE` directive.

Options: none.
Parameters: none.

See also
Directives: `CASE`, `OR`, `ELSE`, `EXIT`.
Genstat Reference Manual 1 Summary section on: Program control.

ENDDEBUG
Cancels a DEBUG statement.

No options or parameters

Description
ENDDEBUG ends a debugging session instigated by the DEBUG directive, where more details are given.

Options: none.
Parameters: none.

See also
Directives: DEBUG, BREAK.
Genstat Reference Manual 1 Summary section on: Program control.

ENDFOR
Indicates the end of the contents of a loop.

No options or parameters

Description
Loops are introduced by the FOR directive, where full details are given.

Options: none.
Parameters: none.

See also
Directives: FOR, EXIT.
Genstat Reference Manual 1 Summary section on: Program control.

ENDIF

Indicates the end of a block-if control structure.

No options or parameters

Description

A *block-if* structure consists of one or more alternative sets of statements. The first of these is introduced by an IF statement. There may then be further sets introduced by ELSIF statements. Then you can have a final set introduced by an ELSE statement, and the whole structure is terminated by an ENDIF statement. Full details are given in the description of the IF directive.

Options: none.
Parameters: none.

See also

Directives: IF, ELSIF, ELSE, EXIT.
Genstat Reference Manual 1 Summary section on: Program control.

ENDJOB
Ends a Genstat job.

No options or parameters

Description
A Genstat program can be split up into individual *jobs*. These can each be terminated by the ENDJOB directive. Full details are given in the description of the JOB directive, which can be used to start a new job.

Options: none.
Parameters: none.

See also
Directives: JOB, STOP.
Genstat Reference Manual 1 Summary section on: Program control.

ENDPROCEDURE

Indicates the end of the contents of a Genstat procedure.

No options or parameters

Description

ENDPROCEDURE ends the definition of a Genstat *procedure*. Full details are given in the description of the PROCEDURE directive

Options: none.
Parameters: none.

See also

Directives: PROCEDURE, OPTION, PARAMETER, CALLS.
Genstat Reference Manual 1 Summary section on: Program control.

ENQUIRE

Provides details about files opened by Genstat.

No options

Parameters

CHANNEL = *scalars*	Channel numbers to enquire about; for FILETYPE=input or output, a scalar containing a missing value will be set to the number of the current channel of that type and a negative value can be used to check the existence of a file that is not yet connected to a channel
FILETYPE = *string tokens*	Type of each file (input, output, unformatted, backingstore, procedurelibrary, graphics); default inpu
OPEN = *scalars*	To indicate whether or not the corresponding channels are currently open (0=closed, 1=open)
NAME = *texts*	External name of the file, if channel is open
EXIST = *scalars*	To indicate whether files on corresponding channels currently exist (0=not yet created, 1=exist)
WIDTH = *scalars*	Maximum width of records in each file (only relevant for input and output files, set to * for other types)
PAGE = *scalars*	Number of lines per page (relevant only for output files)
ACCESS = *texts*	Allowed type of access: set to 'readonly', 'writeonly' or 'both'
LINE = *scalars*	Number of the current line (input files only)
STYLE = *texts*	Underlying style of an output channel: set to 'plaintext', 'html', 'rtf', or 'latex' (see OPEN)
OUTSTYLE = *texts*	Current style of an output channel: set to 'plaintext' or 'formatted' (see OUTPUT)
SIZE = *texts*	Size of the file, in bytes

Description

ENQUIRE allows you to ascertain whether a particular channel is already in use and, if so, what properties are defined for aspects like the width of each line or the number of lines per page (see the OPEN and OUTPUT directives). This is likely to be of most use within general programs and procedures.

You specify the channel using the parameters CHANNEL and FILETYPE; the other parameters allow you to save the required information in data structures of the appropriate type.

ENQUIRE can also be used to discover whether a file exists. You simply set the CHANNEL option to a negative number. The result of the enquiry is saved by the EXIST parameter. So, for example

```
ENQUIRE CHANNEL=-1; NAME='lost.dat'; EXIST=Found
```

will set the scalar Found to one if the file lost.dat exists, or to zero otherwise. Similarly, the SIZE parameter can provide the size of the file (in bytes): for example

```
ENQUIRE CHANNEL=-1; NAME='lost.dat'; SIZE=Nbytes
```

A missing value is returned if the file does not exist.

Options: none.
Parameters: CHANNEL, FILETYPE, OPEN, NAME, EXIST, WIDTH, PAGE, ACCESS, LINE, STYLE, OUTSTYLE.

See also
Directives: OPEN, CLOSE, FCOPY, FDELETE, FRENAME.
Genstat Reference Manual 1 Summary section on: Input and output.

EQUATE

Transfers data between structures of different sizes or types (but the same modes i.e. numerical or text) or where transfer is not from single structure to single structure.

Options

`OLDFORMAT` = *variate*	Format for values of `OLDSTRUCTURES`; within the variate, a positive value *n* means take *n* values, −*n* means skip *n* values and a missing value means skip to the next structure; default `*` i.e. take all the values in turn
`NEWFORMAT` = *variate*	Format for values of `NEWSTRUCTURES`; within the variate, a positive value *n* means fill the next *n* positions, −*n* means skip *n* positions and a missing value means skip to the next structure; default `*` i.e. fill all the positions in turn
`FREPRESENTATION` = *string token*	How to interpret factor values (`labels`, `levels`, `ordinals`); default `leve`

Parameters

`OLDSTRUCTURES` = *identifiers*	Structures whose values are to be transferred; if values of several structures are to be transferred to one item in the `NEWSTRUCTURES` list, they must be placed in a pointer
`NEWSTRUCTURES` = *identifiers*	Structures to take each set of transferred values; if several structures are to receive values from one item in the `OLDSTRUCTURES` list, they must be placed in a pointer

Description

The `EQUATE` directive copies values from one set of data structures to another. For example, you may wish to copy the values from a one-way table into a variate, or from a matrix into a set of variates (one variate for each row, or for each column), or the other way round, from variates into a matrix. Alternatively, you may want to append values from several data structures into a single one. The only constraint is that the structures in the respective sets must all contain the same kind of values.

The general idea with `EQUATE` is that the values in the structures in the `OLDSTRUCTURES` list are copied into the structures in the `NEWSTRUCTURES` list. Each item in `OLDSTRUCTURES` list specifies a single data structure, or a single set of data structures, containing the values to be transferred. A single structure can be a factor, or a text, or any one of the structures that contain numbers (scalar, variate, rectangular matrix, diagonal matrix, symmetric matrix or table). If you want to give a set of structures you must put them into a pointer. As already mentioned, all the structures in the set must contain the same kind of values: that is, they must all be texts, or all factors, or must all contain numbers (but they need not all be the same kinds of numerical structure – they could, for example, be a mixture of variates and matrices).

The corresponding entry in the `NEWSTRUCTURE` list indicates where the transferred data are to be placed. It is either a single structure or a pointer to a set of structures; the structures must be of a type suitable to store the values to be transferred.

Except with a format Genstat ignores where each structure within a set from the `OLDSTRUCTURES` list ends and another one begins: that is, it treats the set as being a concatenated list of values. Similarly, it treats the structures in each `NEWSTRUCTURES` set as an unstructured list of positions that are to receive values. The old values are repeated as often as is necessary to traverse all the new positions.

You can use the OLDFORMAT and NEWFORMAT options to control how the old values and new positions are traversed. The setting for each of these is a variate whose values are interpreted as follows:

(a) a positive integer *n* means take the next *n* values (OLDFORMAT) or fill the next *n* positions (NEWFORMAT);
(b) a negative integer −*n* means skip the next *n* values or positions;
(c) a missing value * means skip to the end of the structure.

As usual, Genstat recycles when it runs out of values. That is, if the contents of one of the variates is exhausted before all the NEWSTRUCTURES positions have either been filled or skipped, then that variate is repeated.

If you are transferring values between factors, Genstat will check that each value to be transferred is valid for the factor in the NEWSTRUCTURES list. By default, Genstat will try to match the values using the levels of the factors, but you can set option FREPRESENTATION=labels to match by their labels, or FREPRESENTATION=ordinals to match them merely according to the ordinal position in the levels vector of each factor.

The values of factors that have labels can be copied into texts. In addition, values of texts can be copied into factors, provided all the strings are valid labels for the factor concerned.

Factor values can also be copied into variates; the FREPRESENTATION option controls whether Genstat uses the levels or the ordinal values.

Options: OLDFORMAT, NEWFORMAT, FREPRESENTATION.
Parameters: OLDSTRUCTURES, NEWSTRUCTURES.

Action with RESTRICT
Any restrictions on vectors in an EQUATE statement are ignored.

See also
Directive: CALCULATE.
Procedures: APPEND, STACK, UNSTACK, SUBSET, VEQUATE.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

ESTIMATE

Estimates parameters in Box-Jenkins models for time series (synonym of TFIT).

Options

PRINT = *string tokens*	What to print (model, summary, estimates, correlations, monitoring); default mode,summ,esti
LIKELIHOOD = *string token*	Method of likelihood calculation (exact, leastsquares, marginal); default exac
CONSTANT = *string token*	How to treat the constant (estimate, fix); default esti
RECYCLE = *string token*	Whether to continue from previous estimation (yes, no); default no
WEIGHTS = *variate*	Weights; default *
MVREPLACE = *string token*	Whether to replace missing values by their estimates (yes, no); default no
FIX = *variate*	Defines constraints on parameters (ordered as in each model, tf models first): zeros fix parameters, parameters with equal numbers are constrained to be equal; default *
METHOD = *string token*	Whether to carry out full iterative estimation, to carry out just one iterative step, to perform no steps but still give parameter standard deviations, or only to initialize for forecasting by regenerating residuals (full, onestep, zerostep, initialize); default full
MAXCYCLE = *scalar*	Maximum number of iterations; default 15
TOLERANCE = *scalar*	Criterion for convergence; default 0.0004
SAVE = *identifier*	To name save structure, or supply save structure with transfer-functions; default * i.e. transfer-functions taken from the latest model

Parameters

SERIES = *variate*	Time series to be modelled (output series)
TSM = *TSM*	Model for output series
BOXCOXMETHOD = *string token*	How to treat transformation parameter in output series (fix, estimate); default fix
RESIDUALS = *variate*	To save residual series

Description

ESTIMATE was renamed as TFIT in Release 14 to emphasize its status as a time-series command. The earlier name (ESTIMATE) was retained to allow previous programs to continue to run, but this may be removed in a future release.

The main use of ESTIMATE is to fit parameters to time-series models, although you can also use it to initialize for the FORECAST directive, even when the model parameters are already known. You need to define a TSM structure beforehand, for use as input to the TSM parameter. You may also wish to give a TRANSFERFUNCTION statement for example if you wish to specify explanatory variables for regression with ARIMA errors or to define transfer-function models. In many applications of estimating a univariate ARIMA model, you will need only a simple form of the directive, such as:

```
ESTIMATE Daylength; TSM=Erp
```

The SERIES parameter specifies the variate holding the time series data to which the model

is to be fitted.

The TSM parameter specifies the ARIMA model that is to be fitted to the time-series data. This TSM must already have been declared and its ORDERS must have been set. If the LAGS parameter of the TSM has been set, the lags must have been given values. However, if the PARAMETERS of the TSM model have been set, these need not have been declared previously nor given values. When the parameter values are not set, default values are used: these are all zero, except for the transformation parameter, which is set to 1.0 if it is not to be estimated (see BOXCOXMETHOD and FIX below). Any parameter values that you do specify will be used as initial values for the parameters in the model; Genstat replaces any missing values by the default values. If any group of autoregressive or moving-average parameters do not satisfy the required conditions for stationarity or invertibility, all the parameters to be estimated are reset by Genstat to the default values. After ESTIMATE, the parameters of the TSM contain the estimated parameter values.

The BOXCOXMETHOD parameter allows you to estimate the transformation parameter λ.

The RESIDUALS parameter saves the estimated innovations (or residuals). The residuals are calculated for $t=t_0...N$, where $t_0=1+p+d-q$ for a simple ARIMA model. If $t_0>1$, missing values will be inserted for $t=1...t_0-1$.

The PRINT option controls printed output. If you specify monitoring, then at each cycle of the iterative process of estimation, Genstat prints the *deviance* for the current fitted model, together with the current estimates of model parameters. The format is simple with the minimum of description, to let you judge easily how quickly the process is converging. The other settings of PRINT control output at the end of the iterative process. If you specify model, the model is briefly described, giving the identifier of the series and the time-series model, together with the orders of the model. If you specify summary, the deviance of the final model is printed, along with the residual number of degrees of freedom. If you specify estimates, the estimates of the model parameter are printed in a descriptive format, together with their estimated standard errors and reference numbers. If you specify correlations, the correlations between estimates of parameters are printed, with reference numbers to identify the parameters.

The LIKELIHOOD option specifies the criterion that Genstat minimizes to obtain the estimates of the parameters: this is described in the next section. The default setting exact is recommended for most applications.

You can use the CONSTANT option to specify whether Genstat is to estimate the constant term c in the model. If CONSTANT=fix, the constant is held at the value given in the initial parameter values; this need not be zero.

The RECYCLE option allows a previous ESTIMATE statement to continue; this can save computing time. If RECYCLE=yes, the most recent ESTIMATE statement is continued, unless the SAVE option has been set to the save structure from some other ESTIMATE statement. The SERIES and TSM settings are then taken from this previous ESTIMATE statement: Genstat ignores any specified in the current statement. Most of the settings of other parameters and options are carried over from the previous statement, and new values are ignored. However, there are some exceptions. You can change the RESIDUALS variate, you can reset MAXCYCLE to the number of further iterations you require, and you can change the settings of TOLERANCE and PRINT. You can also change the values of the variate in the WEIGHTS option; you can thus get reweighted estimation. You can change the values of the SERIES itself, although you cannot change missing values; if the MVREPLACE option was previously set to yes, you must put the original missing values back into the SERIES variate before the new ESTIMATE statement.

The WEIGHTS option includes in the likelihood a weighted sum-of-squares term

$$\sum_{t=t0\ ...\ N} \{ w_t a_t^2 \}$$

where w_t, $t=1...N$ are provided by the WEIGHTS variate. The values of w_t must be strictly positive. If $t_0<1$, where $t_0=1+d+p-q$, then w_t is taken as 1 for $t<1$.

The MVREPLACE option allows you to request any missing values in the time-series to be replaced by their estimates after estimation. Genstat will always estimate the missing values,

irrespective of the setting of MVREPLACE; so you can also obtain these estimates later from TKEEP.

The FIX option allows you to place simple constraints on parameter values throughout the estimation. The units of the FIX variate correspond to the parameters of the TSM, excluding the innovation variance. The values of the FIX variate are used to define the parameter constraints and must be integers. If an element of the FIX variate is set to 0, the corresponding parameter is constrained to remain at its initial setting. If an element is not 0, and the value is unique in the FIX variate, the parameter is estimated without any special constraint. If two or more values are equal, the corresponding parameters are constrained to be equal throughout the estimation. The number that you give to a parameter by FIX will appear as the reference number of the parameter in the printed model and correlation matrix. This option overrides any setting of CONSTANT and BOXCOXMETHOD.

The MAXCYCLE option specifies the maximum number of iterations to be performed.

The TOLERANCE option specifies the convergence criterion. Genstat decides that convergence has occurred if the fractional reduction in the deviance in successive iterations is less than the specified value, provided also that the search is not encountering numerical difficulties that force the step length in the parameter space to be severely limited. You can use monitoring to judge whether, for all practical purposes, the iterations have converged. Genstat gives warnings if the specified number of iterations is completed without convergence, or if the search procedure fails to find a reduced value of the deviance despite a very short step length. Such an outcome may be due to complexities in the likelihood function that make the search difficult, but can be due to your specifying too small a value for TOLERANCE.

The SAVE option allows you to save the *time-series save structure* produced by ESTIMATE. You can use this in further ESTIMATE statements with RECYCLE=yes, or in FORECAST statements. It can also be used by the TDISPLAY and TKEEP directives. Genstat automatically saves the structure from the most recent ESTIMATE statement, but this is over-written when the next ESTIMATE statement is executed, unless you have used SAVE to give it an identifier of its own. You can access the current time-series save structure by the SPECIAL option of the GET directive, and reset it by the TSAVE option of the SET directive.

The METHOD option has four possible settings. The default setting is full which gives the usual estimation to convergence or until the maximum number of iterations has been reached.

With the setting METHOD=initialize, ESTIMATE carries out only the residual regeneration steps (that is, calculation of a_t for $t=t_0...N$) which are needed before FORECAST can be used. If the model has just been estimated using the default full setting, this is unnecessary. The setting initialize is useful when the time series is supplied with a known model and a minimal amount of calculation is wanted to prepare or initialize for forecasting. None of the model parameters are changed, and no standard errors of parameter estimates are available. Missing values in the series *are* estimated so this setting provides an efficient way of getting their values when the time series model is known; they can then be obtained using TKEEP. The deviance value is also available from TKEEP. This setting is therefore useful for efficient calculation of deviance values when you want to plot the shape of the deviance as a function of parameter values.

With the setting METHOD=zerostep the effect is the same as for initialize except that ESTIMATE also calculates the standard errors of the parameters as if they had just been estimated. These can be used together with other quantities available from TKEEP to construct confidence intervals and carry out tests on the parameter values, which remain unchanged except that the innovation variance in the ARIMA model is replaced by its estimate conditional on all other parameters.

The setting METHOD=onestep gives the same results as specifying the option MAXCYCLE=1 in ESTIMATE. It is convenient for carrying out quick tests of model parameters.

To explain the LIKELIHOOD option, we need to describe the estimation of ARIMA models

in more detail. You may want to skip this if you are doing fairly routine work.

The first step in deriving the likelihood for a simple model is to calculate

$$w_t = \nabla^d y_t - c\,, \qquad t = 1+d \ldots N$$

This has a multivariate Normal distribution with dispersion matrix $V\sigma_a^2$, where V depends only on the autoregressive and moving-average parameters. The likelihood is then proportional to

$$\{ \sigma_a^{2m} \left| V \right| \}^{-½} \exp\{ -w'V^{-1}w/2\sigma_a^2 \}$$

where $m=N-d$. In practice Genstat evaluates this by using the formula

$$w'\, V^{-1}\, w \;=\; W + \sum_{t=t0 \ldots N} \{ a_t^2 \} \;=\; S$$

where $t_0=1+d+p-q$. The term W is a quadratic form in the p values $w_{1+d-q} \ldots w_{p+d-q}$: it takes account of the starting-value problem for regenerating the innovations a_t, and avoids losing information as would happen if the process used only a conditional sum-of-squares function. If $q>0$, Genstat introduces unobserved values of $w_{1+d-q} \ldots w_d$ in order to calculate the sum S. Genstat uses linear least-squares to calculate these q starting values for w, thus minimizing S. We shall call them *back-forecasts*, though if $p>0$ they are actually computationally convenient linear functions of the proper back-forecasts. We shall call S the sum-of-squares function: it is the sum of the quadratic form and the sum-of-squares term, and is identical to the value expressed by Box & Jenkins (1970) as

$$\sum_{t=-\infty \ldots N} \{ a_t^2 \}$$

using infinite back-forecasting; that is, using:

$$W \;=\; \sum_{t=-\infty \ldots t0-1} \{ a_t^2 \}$$

The values a_t for $t=t_0 \ldots N$ agree precisely with those of Box and Jenkins.

To clarify all this, consider examples with no differencing; that is, $d=0$. If $p=0$ and $q=1$ then $W=0$ and $t_0=0$, and one back-forecast w_0 is introduced. If $p=1$ and $q=0$ then $W=(1-\varphi_1^2)w_1^2$ and $t_0=2$, and no back-forecasts are needed. If $p=q=1$ then $W=(1-\varphi_1^2)w_0^2$ and $t_0=1$, and so one back-forecast w_0 is needed. In this case the proper back-forecast is in fact $w_0/(1-\theta_1\varphi_1)$.

The value of $\left| V \right|$ is a by-product of calculating W and the back-forecast. For example, if $p=0$ and $q=1$, then

$$\left| V \right| = (1 + \theta_1^2 + \ldots + \theta_1^{2N})$$

If $p=1$ and $q=0$,

$$\left| V \right| = 1 / (1 - \varphi_1^2)$$

and if $p=q=1$,

$$\left| V \right| = 1 + (\varphi_1 - \theta_1)^2 (1 + \theta_1^2 + \ldots + \theta_1^{2N-2}) / (1 - \varphi_1^2)$$

Concentrating the likelihood over σ_a^2 by setting $\sigma_a^2=S/m$ yields a value proportional to $\{ \left| V \right|^{1/m} S \}^{-m/2}$.

The default setting of the `LIKELIHOOD` option is `exact`. In this case the concentrated likelihood is maximized, by minimizing the quantity

$$D = \left| V \right|^{1/m} S$$

which is called the deviance.

The setting `leastsquares` specifies that Genstat is to minimize only the sum-of-squares term S. This criterion corresponds to the back-forecasting sum-of-squares used by Box & Jenkins (1970), and will in many cases give estimates close to those of the exact likelihood. However, some discrepancy arises if the series is short or the model is close to the invertibility boundary. This is because of limitations on the back-forecasting procedure, as described in the algorithms of Box & Jenkins (1970). The deviance value D that Genstat prints is, with this setting, simply S.

When you use exact likelihood, the factor $\left| V \right|^{1/m}$ reduces bias in the estimates of the parameter; you would get bias if you used `leastsquares` instead. However, $\left| V \right|^{1/m}$ is generally close to one, unless the series is short or the model is either seasonal or close to the boundaries of invertibility or stationarity. The `leastsquares` setting is therefore adequate for most long, non-seasonal sets of data; using it may reduce the computation time by up to 50%. When you specify that Genstat is to estimate the parameter λ of the Box-Cox transformation, Genstat also

includes the Jacobian of the transformation in the likelihood function. The result is an extra factor $G^{-2(\lambda-1)}$ in the definition of the deviance, G being the geometric mean of the data,

$$G = (\prod_{t=1 \ldots N} \{ y_t \}) ** (1 / N)$$

Note that this is not included unless λ is being estimated, even if $\lambda \neq 1$.

You can treat differences in $N\log(D)$ as a chi-square variable in order to test nested models: this is supported by asymptotic theory, and by experience with models that have moderately large sample sizes. Similarly, you can select between different models by using $N\log(D)+2k$ as an information criterion, k being the number of estimated parameters. But both of these test procedures are questionable if the estimated models are close to the boundaries of invertibility or stationarity. Provided all the models that are being compared have the same orders of differencing, with the differenced series being of length m, it is recommended that $m\log(D)$ be used rather than $N\log(D)$ in these tests since $m\log(D)$ is precisely minus two multiplied by the log-likelihood as defined above.

The setting `marginal` is relevant mainly when `ESTIMATE` is used for regression with ARIMA errors. (This requires a `TRANSFERFUNCTION` statement beforehand to specify the explanatory variables.) The likelihood for the model is defined as that of the univariate error series e_t which is defined in general by

$$e_t = y_t - b_1 x_{1,t} - \ldots - b_m x_{m,t}$$

(the x_i being m explanatory variables). The constant term therefore appears in the model after any differencing of e_t; for example

$$\nabla e_t = c + (1 - \theta_1 B) a_t$$

You can get bias in the estimates of the parameters of an ARIMA model because the regression is estimated at the same time. You can guard against this by specifying `LIKELIHOOD=marginal`. This can be particularly important if the series are short or if you use many explanatory variables (Tunnicliffe Wilson 1989). The deviance is now defined as

$$D = S (\left| X' V^{-1} X \right| \left| V \right|)^{1/m}$$

where m is reduced by the number of regressors (including the constant term) and the columns of X are the differenced explanatory series: the other terms are as in the exact likelihood.

You can use the `marginal` setting also for univariate ARIMA modelling, when the constant term is the only explanatory term. Furthermore, Genstat deals with missing values in the response variate by doing a regression on indicator variates; these too are included in the X matrix. However, you cannot use marginal likelihood and estimate a transformation parameter in either the transfer-function model or an ARIMA model. Neither can you use it if you set the `FIX` option in `ESTIMATE`. In these cases Genstat automatically resets the `LIKELIHOOD` option to `exact`.

At every iteration with the setting `LIKELIHOOD=marginal`, the regression coefficients are the maximum-likelihood estimates conditional upon the estimated values of the parameters of the ARIMA model: these are also the generalized least-squares estimates, conditioned in the same way. This is so even if `MAXCYCLE=0`; that is, the coefficients of the regression are re-estimated even at iteration 0. Therefore you must not use the `marginal` setting with the option `METHOD=initialize` to initialize for `FORECAST`. You can compare deviance values that were obtained using marginal likelihood only for models with the same explanatory variables and the same differencing structure in the error model.

Options: `PRINT`, `LIKELIHOOD`, `CONSTANT`, `RECYCLE`, `WEIGHTS`, `MVREPLACE`, `FIX` , `METHOD`, `MAXCYCLE`, `TOLERANCE`, `SAVE`.

Parameters: `SERIES`, `TSM`, `BOXCOXMETHOD`, `RESIDUALS`.

Action with `RESTRICT`

The `SERIES` variate can be restricted, but this must be to a contiguous set of units.

References

Box, G.E.P. & Jenkins, G.M. (1970). *Time Series Analysis, Forecasting and Control*. Holden-Day, San Francisco.

Tunnicliffe Wilson, G. (1989). On the use of marginal likelihood in time-series model estimation. *Journal of the Royal Statistical Society, Series B*, **51**, 15-27.

See also

Directive: `TFIT`.

Genstat Reference Manual 1 Summary section on: Time series.

EXECUTE
Executes the statements contained within a text.

No options

Parameter

texts	Statements to be executed

Description
EXECUTE allows a set of Genstat statements placed in a text to be executed, for example inside loops or procedures.

Options: none.
Parameter: unnamed.

Action with RESTRICT
Any restrictions on the texts are ignored.

See also
Directives: TEXT, PROCEDURE, FOR.
Genstat Reference Manual 1 Summary section on: Program control.

EXIT

Exits from a control structure.

Options

NTIMES = *scalar*	Number of control structures, *n*, to exit (if *n* exceeds the number of control structures of the specified type that are currently active, the exit is to the end of the outer one; while for *n* negative, the exit is to the end of the −*n*'th structure in order of execution); default 1
CONTROLSTRUCTURE = *string token*	Type of control structure to exit (job, for, if, case, procedure); default for
REPEAT = *string token*	Whether to go to the next set of parameters on exit from a FOR loop or procedure (yes, no); default no
EXPLANATION = *text*	Text to be printed if the exit takes place; default *

Parameter

expression	Logical expression controlling whether or not an exit takes place

Description

Sometimes you may want simply to abandon part of a program: you may be unable to do any further calculations or analyses. For example, if you are examining several subsets of the units, you would wish to abandon the analysis of any subset that turned out to contain no observations. Another example would be if you wanted to abandon the execution of a procedure whenever an error diagnostic has appeared. The EXIT directive allows you to exit from any control structure.

In its simplest form EXIT has no parameter setting, and the exit is unconditional: Genstat will always exit from the control structure or structures concerned. You are most likely to use this as part of an ELSE block of a block-if or multiple-selection structure. For example

```
IF N.GT.0
  CALCULATE Percent = R * 100 / N
ELSE
  PRINT [IPRINT=*] 'Incorrect value ',N,'  for N.'
  EXIT [CONTROLSTRUCTURE=procedure]
ENDIF
```

prints an appropriate warning message for a zero or negative value of N, and then exits from a procedure.

If the warning message is simply a text or string, the EXPLANATION option can be used to print it on exit. For example

```
EXIT [CONTROLSTRUCTURE=procedure; \
  EXPLANATION='Incorrect value for N.'] N.LE.0
CALCULATE Percent = R * 100 / N
```

has the same effect except that the actual value of N is no longer printed.

The CONTROLSTRUCTURE option specifies the type of control structure from which to exit. The default setting is for, causing an exit from a FOR loop. For the other settings: if causes an exit from a block-if structure (as introduced by the IF directive), case exits from a multiple-selection structure (as introduced by CASE), procedure exits from a procedure (see the PROCEDURE directive), and job causes the entire job to be abandoned (see JOB). Sometimes, to exit from one type of control structure, others must be left too. To exit from the procedure in the above example, requires Genstat to exit also from the block-if structure. Generally, Genstat does these nested exits automatically, as required. However, inside a procedure, you can exit only from FOR loops and block-if or multiple-selection structures that are within the procedure. You

cannot put, for example,

```
EXIT [CONTROLSTRUCTURE=if]
```

within a part of the procedure where there is no block-if in operation, and then expect Genstat to exit both from the procedure and from a block-if structure in the outer program from which the procedure was called. Genstat regards a procedure as a self-contained piece of program.

The NTIMES option indicates how many control structures of the specified type to exit from. If you ask Genstat to exit from more structures than are currently in operation in your program, it will exit from as many as it can and then print a warning. If NTIMES is set to zero or to missing value no exit takes place. If NTIMES is set to a negative value, say $-n$, the exit is to the end of the nth structure of the specified type, counting them in the order in which their execution began. Consider this example:

```
FOR I=A[1...3]
  FOR J=B[1...3]
    FOR K=C[1...3]
      FOR L=D[1...3]
        "contents of the inner loop, including:"
        EXIT [NTIMES=Nexit]
        "amongst other statements"
      ENDFOR "end of the loop over D[]"
    ENDFOR "end of the loop over C[]"
  ENDFOR "end of the loop over B[]"
ENDFOR "end of the loop over A[]"
```

If the scalar Nexit has the value 2, the exit is to the end of the loop over C[]; so the two exits are from the loop over D[] and the loop over C[]. But if Nexit has the value -2 the exit is to the end of the loop over B[], as this is the second loop to have been started.

A further possibility when EXIT is used within a FOR loop is that you can choose either to go right out of the loop and continue by executing the statement immediately after the ENDFOR statement, or to go to ENDFOR and then repeat the loop with the next set of parameter values. To repeat the loop, you need to set option REPEAT=yes. For example, suppose that variates Height and Weight contain information about children of various ages, ranging from five to 11. The RESTRICT statement causes the subsequent GRAPH statement to plot only those units of Height and Weight where the variate Age equals Ageval. The EXIT statement ensures that the graph is not plotted if there are no units of a particular age; the program then continues with Ageval taking the next value in the list.

```
FOR Ageval=5,6,7,8,9,10,11
  RESTRICT Height,Weight; CONDITION=Age.EQ.Ageval
  EXIT [REPEAT=yes] NVALUES(Height).EQ.0
  GRAPH Height; X=Weight
ENDFOR
```

The REPEAT option can also be used within a procedures to ask Genstat to call the procedure with the next set of parameter settings.

The example of the heights and weights of children also illustrates the use of the parameter of EXIT, to make the effect conditional. The parameter is an expression which must evaluate to a single number which Genstat interprets as a logical value. If the value is zero, the condition is *false* and no exit takes place; for other values the condition is *true* and the exit takes effect as specified. This is particularly useful for controlling the convergence of iterative processes: for example

```
CALCULATE Clim = X/10000
FOR [NTIMES=999]
  CALCULATE Previous = Root
  & Root = (X/Previous + Previous)/2
  PRINT Root,Previous; DECIMALS=4
  EXIT ABS(Previous-Root) < Clim
ENDFOR
```

will calculate the square root of `X` to four significant figures.

Options: `NTIMES`, `CONTROLSTRUCTURE`, `REPEAT`, `EXPLANATION`.
Parameter: unnamed.

See also
Directives: `FOR`, `CASE`, `IF`, `FAULT`, `CALCULATE`.
Genstat Reference Manual 1 Summary section on: Program control.

EXPRESSION

Declares one or more expression data structures.

Options

VALUE = *expression*	Value for all the expressions; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the expressions in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the expressions
VALUE = *expression structures*	Expression data structures providing values for the expressions
EXTRA = *texts*	Extra texts associated with the identifiers

Description

The IDENTIFIER parameter lists the identifiers of the expressions that are to be declared. The expression data structure stores a Genstat expression, for example

```
Hours = Minutes/60
```

Usually you will find it easiest to type out an expression like this explicitly whenever you need it. The main use, then, for this rather specialized data structure is to supply an expression as the argument of a procedure.

Values can be assigned to the expressions by either the VALUE option or the VALUE parameter. The option defines a common value for all the structures in the declaration, while the parameter allows the structures each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

You can associate a text of extra annotation with each expression using the EXTRA parameter. If MODIFY is set to yes any existing attributes and values of the expressions are retained; otherwise these are lost.

Here are two examples using the VALUE option:

```
EXPRESSION [VALUE=Length*Width*Height] Vcalc
EXPRESSION [VALUE=Dose=LOG10(Dose)] Dtrans
```

These put the expression Length*Width*Height into the identifier Vcalc, and the expression Dose=LOG10(Dose) into Dtrans. Both expressions could be declared simultaneously, using the VALUE parameter, by putting

```
EXPRESSION Vcalc,Dtrans; VALUE=!E(Length*Width*Height), \
  !E(Dose=LOG10(Dose))
```

!E(Length*Width*Height), for example, is an unnamed expression.

The IPRINT option can be set to specify how the expressions will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

Options: VALUE, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUE, EXTRA.

See also
Directives: CALCULATE, FARGUMENTS, SETCALCULATE.
Procedure: DFUNCTION.
Genstat Reference Manual 1 Summary sections on: Data structures, Calculations and manipulation.

EXTERNAL

Declares an external function in a DLL for use by the OWN function.

Options

LIBRARY = *text*	Name of DLL file containing the function

Parameters

FUNCTION = *text*	Name of the function entry point in the DLL
NAME = *text*	Name for the function to be used in the OWN function; default uses the name set in FUNCTION
RESULTS = *string token*	The type of result returned from the function (summary, transformation); default tran
NPARAMETERS = *scalar*	The number of parameters in the function call; default 0
ERRORS = *scalar* or *variate*	Error codes returned from the function; default * i.e. no error codes
MESSAGES = *text*	Messages for the corresponding error codes

Description

Genstat can execute external functions, stored in DLLs, using the OWN function. You first need to create the DLLs by compiling Fortran or C programs. You can then use EXTERNAL to define the link to Genstat. For example:

```
EXTERNAL [LIBRARY='CurveFuncs.dll'] FUNCTION='PCNORMAL'; NPAR=0
CALCULATE EX = OWN(X; 'PCNORMAL')
EXTERNAL [LIBRARY='CurveFuncs.dll'] FUNCTION='PEAKRISE'; NPAR=1
CALCULATE P = OWN(X; 'PEAKRISE'; 10)
EXTERNAL [LIBRARY='CurveFuncs.dll'] FUNCTION='HWA';\
   RESULTS=summary; NPAR=5
CALCULATE SS = OWN(X; 'HWA'; a; b; g; n; s)
```

defines three functions in CurveFuncs.dll and then uses these in calculations. The first two return variates with the same length as the first argument X, and the third returns a scalar. The functions have zero, one and five parameters respectively. The parameters in the OWN function follow the data and name arguments, and must be scalars.

The LIBRARY option specifies the name of the file. If the full path to the DLL is not provided, the user add-ins folder is searched first. If the file is not found there, the system add-ins folder is searched. An FI 11 fault is generated if the file is not found.

The FUNCTION parameter gives the name of the entry point in the DLL for the function, and is case insensitive. The entry point must be exported when the program library is compiled. For example, in a C program the function declaration should contain __declspec(dllexport) or its equivalent. If the function entry point is not found in the program library, an FI 12 fault is generated. If you wish to refer to the function in the OWN function by a different name to its entry-point name, you can define that name with the NAME parameter.

The RESULTS parameter indicates what type of result is returned: summary returns a scalar and transformation (default) returns a structure of the same type and size as the first argument.

The NPARAMETERS parameter defines the number of scalar parameters that follow the function name in the OWN function. By default there are none.

The ERRORS, and MESSAGES parameters can be used to set up user-defined fault codes and text for the corresponding error messages for the external functions. If you are using several external functions in a program, you should use different error codes in the different functions, unless the meaning of the code is common to all functions, as the error code/message table is combined over all functions. This allows you to specify just one set of error codes/messages over a set of functions in a library. So, for example, all functions could return a common code 9 if

they run out of memory.

The function declaration in C takes the form:

```
long NAME(double* X, int* NX, double* P, int* NP, double* R, int* NR)
```

where

- X is an array of the input data from the first argument in the OWN function,
- NX is the number of elements in X,
- P is a pointer to an array of the parameters in OWN function,
- NP is the number of parameters,
- R is a pointer to the array to hold the results, and
- NR is the number of elements in the result array.

NR must be 1 if RESULTS = summary and NX otherwise. The passed array P is NP+1 long, and the last element is the value that Genstat uses to represent a missing value. The function should return zero if it completes successfully, and a positive error code if there is a fault. The error code will be translated to the text in MESSAGES if the error code matches one set up by ERRORS.

For a Fortran program, the declaration would be:

```
INTEGER FUNCTION NAME(X,NX,P,NP,R,NR)
   INTEGER   NX,NP,NR
   REAL*8    X(*),P(*),R(*)
   !DIR$ ATTRIBUTES REFERENCE X,NX,P,NP,R,NR
```

There are example programs in Fortran (OwnFunction.f90) and C (OwnFunction.c) in the Genstat Source folder. These can be compiled to a DLL library using the appropriate compiler and linker settings. A DLL created from these (OwnFunction.dll) is included in the Genstat system AddIns folder should you want to test this. The function OwnFunction in this DLL calculates the sine of an argument in degrees rather than the usual radians in the standard SIN function. Note, however that these simple examples do not test for missing values in the input data.

Any numeric structure (scalar, variate, matrix, symmetric matrix or diagonal matrix) can be passed to the OWN function and, when RESULTS = transformation, the returned result will be the same type and size of structure (e.g. passing a 3 by 4 matrix will return a 3 by 4 matrix). If you need to pass multiple variates to the function, these could be stacked by the APPEND procedure, and the number of variates could be passed as a parameter. Within the function you would extract the stacked values back into 3 arrays and process these individually. If a single variate is to be returned, you would set just the first NX values in R, and then use its first NX values in Genstat to create the resulting variate. For example:

```
EXTERNAL [LIBRARY=VarFunc.dll] FUNCTION='XYZFUNC'; NPAR=1
APPEND [V3] X,Y,Z
CALCULATE V = OWN(V3; 'XYZFUNC'; 3)
CALCULATE N = NVALUES(X)
CALCULATE R = V$[!(1...N)]
```

Option: LIBRARY.
Parameters: FUNCTION, NAME, RESULTS, NPARAMETERS, ERRORS, MESSAGES.

See also
Directives: CALCULATE, PASS, SUSPEND, OWN.
Functions: OWN .
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

FACROTATE

Rotates factor loadings from a principal components, canonical variates or factor analysis.

Options

PRINT = *string tokens*	Printed output required (communalities, loadings, orthogonalrotationmatrix, rotation); default * i.e. no printing
METHOD = *string token*	Criterion (varimax, quartimax); default vari
NROOTS = *scalar*	Sets the number of dimensions to rotate from the original loadings; default * i.e. all

Parameters

OLDLOADINGS = *matrices*	Original loadings
NEWLOADINGS = *matrices*	Rotated loadings for each set of OLDLOADINGS
COMMUNALITIES = *matrices*	Communalities of the variables in each rotation
ROTATION = *matrices*	Saves the orthogonal rotation from the original solution to the rotated space

Description

FACROTATE rotates factor loadings from a principal components, canonical variates or factor analysis (see the PCP, CVA and FCA directives, respectively). The first parameter, OLDLOADINGS, specifies a list of matrices, that contain the loadings for the original dimensions. These can be obtained from the first element of the LRV structures, that can be saved by the LRV parameter of PCP, CVA and FCA. The matrices to save the new loadings are specified by the NEWLOADINGS parameter. The ROTATION parameter can save the orthogonal rotations from the original solutions to the rotated spaces.

Principal components and canonical variates and factor analyses all define a set of dimensions (sometimes called axes) that are linear combinations of the original variables. The individual coefficients of these combinations are called *loadings*, and can be used to interpret the dimensions. With principal components analysis, the loadings must lie in the range [−1, +1]; this is the situation that we discuss initially. The situation with canonical variates and factor analysis is slightly different and is described later.

When several dimensions are considered it is possible to define an equivalent set of new dimensions, whose loadings are linear combinations of the original loadings. If the absolute values of the loadings for a new dimension are either close to 0 or close to 1, you can interpret the dimension as mainly representing only those original variables with large positive (or negative) loadings. You may sometimes want new dimensions determined by loadings like these, because they are easier to interpret. The methods by which these new dimensions can be obtained are generally known collectively as *factor rotation* because the new dimensions represent a rotation of the axes of the original dimensions. The FACROTATE directive provides two methods of orthogonal factor rotation: varimax rotation and quartimax rotation (Cooley & Lohnes 1971). The default method, varimax rotation, maximizes the variance of the squares of the loadings within each new dimension: the effect of this rotation should be to spread out the squared-loadings to the extremes of their range. Quartimax rotation uses the fourth power of the loadings instead of the second power.

Under either method of factor rotation, the total contribution of each of the original variables always remains the same as in the input set of loadings (for mathematical reasons). These contributions are called the *communalities* of the variables, and can be expressed as the sum of the squared loadings: they indicate how much of the variation of each of the original variables is retained in either set of dimensions (whether the original set from the principal component analysis, or the new set from the rotation). They can be saved using the COMMUNALITIES

parameter. If you keep all the loadings from a principal components analysis, each of the variables will have communality 1. Factor rotation in this case will simply give a set of new loadings, each of which will represent just one of the variables, with loading 1. Thus factor rotation is sensible only if you keep merely the higher-dimensional loadings.

The loadings from canonical variates analysis are not constrained to lie in the range [–1, +1]. The factor rotation methods operate in a similar manner as for principal component loadings. Again, the objective is to obtain loading values, such that each is either relatively small or relatively large. Also the communalities of the variables remain the same in the rotated loadings as in the original loadings, and the new loadings are obtained as an orthogonal rotation of the old loadings. However, the complete set of loadings can generally be retained from canonical variate analysis and used for factor rotation, without giving meaningless results. This is because the original dimensions from the canonical variates analysis do not contain all the dimensionality of the original variables, unless the number of variables is less than the number of groups. So a factor rotation of all the dimensions will not merely recover the original variables, as would happen with loadings from principal components analysis. Likewise, loadings from the full set of available dimensions in a factor analysis can be also be retained for rotation without recovering the original variables.

Printed output is controlled by the `PRINT` option, with the following settings:

`communalities`	to print the communalities;
`loadings`	to print the rotated loadings, under the caption "Rotated factors";
`orthogonalrotationmatrix`	to print the rotation matrix;
`rotation`	this is the original setting used to print the rotated loadings. It is retained as a synonym of `loadings` to allow earlier programs to run. However, in view of the confusion with the `ROTATION` parameter, it may be deleted in a future release.

By default, nothing is printed.

The `NROOTS` option sets the number of dimensions to rotate from the original loadings (the other dimensions are left unchanged). The default is to rotate them all.

Options: `PRINT`, `METHOD`, `NROOTS`.
Parameters: `OLDLOADINGS`, `NEWLOADINGS`, `COMMUNALITIES`, `ROTATION`.

Reference

Cooley, W.W. & Lohnes, P.R. (1971). *Multivariate Data Analysis*. Wiley, New York.

See also

Directives: `CVA`, `FCA`, `PCP`, `ROTATE`.
Procedures: `GENPROCRUSTES`, `PCOPROCRUSTES`.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

FACTOR

Declares one or more factor data structures.

Options

NVALUES = *scalar* or *vector*	Number of units, or vector of labels; default * takes the setting from the preceding UNITS statement, if any
LEVELS = *scalar* or *vector*	Number of levels, or series of numbers which will be used to refer to levels in the program; default *
VALUES = *numbers*	Values for all the factors, given as levels; default *
LABELS = *text*	Labels for levels, for input and output; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
REFERENCELEVEL = *scalar*	Defines the reference level used e.g. to define the parameterization of regression models
IPRINT = *string tokens*	Information to be used by default to identify the factors in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the factors
VALUES = *identifiers*	Values for each factor, specified as levels or labels
DECIMALS = *scalars*	Number of decimals for printing levels
CHARACTERS = *scalars*	Number of characters for printing labels
EXTRA = *texts*	Extra text associated with each identifier
DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represent dates and times

Description

Factors are used to indicate groupings of units. The commonest occurrence is in designed experiments. For example, suppose you had 12 observations in an experiment, the first four on one treatment, the next four on a second treatment, and the last four on a third. Then you could record which treatment went with which observation by declaring a factor with the values

```
1,1,1,1,2,2,2,2,3,3,3,3
```

Thus a factor is a vector that has only a limited set of possible values, one for each group; this limitation distinguishes factors from variates and texts. In Genstat, the groups are referred to by numbers known as *levels*. Unless otherwise specified these are the integers 1 up to the number of groups, as in our example; however, you can specify any other numbers by the LEVELS option of the FACTOR directive. You can also give textual labels to the groups, using the LABELS option of FACTOR: these might, for example, be mnemonics for the biochemical names of treatments in an experiment.

Use of the VALUES parameter to assign values has the advantage that you can refer either to labels or to levels; the VALUES option lets you refer only to levels. So, to summarize, the LEVELS and LABELS options list the groups that can occur, while the VALUES option or parameter specifies which groups actually do occur, and in what pattern over the units.

Our simple explanatory example would therefore be:

```
FACTOR [LEVELS=3; VALUES=4(1...3)] Treatment
```

Other examples are:

```
FACTOR [LEVELS=!(2,4,8,16); VALUES=8,4,2,16,4,2,16,8,2] Dose
```

```
FACTOR [LABELS=!T(male,female)] Sex;VALUES=!T(4(male,female))
FACTOR [LEVELS=!(0,2.5,5); LABELS=!T(none,standard,double)]\
  Rate; VALUES=!(0,5,2.5,5,0,2.5)
```

Notice that if we had assigned the values using the VALUES option in the second of these, we would have needed to use the (numerical) levels:

```
FACTOR [LABELS=!T(male,female); VALUES=4(1,2)] Sex
```

Conversely, in the VALUES parameter in the declaration of Rate, we can use either the labels or the levels; so the following statement gives Rate exactly the same values:

```
FACTOR [LEVELS=!(0,2.5,5); LABELS=!T(none,standard,double)]\
  Rate; VALUES=!T(none,double,standard,double,none,standard)
```

The DECIMALS parameter allows you to define a number of decimal places to be used by default when the levels of each factor are printed. The CHARACTERS parameter allows you to define the number of characters to be printed by default when the labels of each factor are printed. You can associate a text of extra annotation with each factor using the EXTRA parameter.

The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each factor to indicate that the factor stores dates and times, and to define a format to be used for these, by default, when they are printed; details are given in the description of the PRINT directive.

In some contexts Genstat needs to treat one of the levels of a factor as a *reference level*. For example, in regression with groups, the parameterization used by Genstat usually involves comparisons of the second and subsequent levels of the grouping factor with its first level. This can be changed by setting the REFERENCELEVEL option to a level other than the first level when the grouping factor is declared.

If the MODIFY option is set to yes any existing attributes and values of the factors are retained if still appropriate; otherwise these are lost.

The IPRINT option can be set to specify how the factors will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

Options: NVALUES, LEVELS, VALUES, LABELS, MODIFY, REFERENCELEVEL, IPRINT.
Parameters: IDENTIFIER, VALUES, DECIMALS, CHARACTERS, EXTRA, DREPRESENTATION.

See also

Directives: GENERATE, GROUPS, VARIATE, TEXT, UNITS.
Procedures: FACAMEND, FACDIVIDE, FACEXCLUDEUNUSED, FACGETLABELS, FACPRODUCT, FACSORT, FACLEVSTANDARDIZE, FACUNIQUE, FMFACTORS, FFREERESPONSEFACTOR, FACCOMBINATIONS, PFACLEVELS, QFACTOR.
Genstat Reference Manual 1 Summary section on: Data structures.

FARGUMENTS

Forms lists of arguments involved in an expression.

Options

EXPRESSION = *expression structure*	Expression whose arguments are required
NRESULTS = *scalar*	Number of results generated by the expression
NCALCULATIONS = *scalar*	Number of calculations in the expression

Parameters

ICALCULATION = *scalars*	The calculation from which to save the result and arguments
RESULT = *dummies*	Stores the result structure for calculation ICALCULATION
ARGUMENTS = *pointers*	Stores the arguments in calculation ICALCULATION

Description

The FARGUMENTS directive allows you to access the data structures involved in a Genstat expression. For example, the expression

```
P = R / N
```

defines a calculation involving the data structures R and N, and stores the results in the data structure P. An expression can contain lists, to define several calculations. For example,

```
P1,P2 = R1,R2 / N1,N2
```

has two results, P1 and P2, that are calculated as R1/N1 and R2/N2 respectively. Some expressions may have no results, for example

```
LOG(P1/(1-P1))
```

This could then be used as part of a CALCULATE statement, as in the following program

```
EXPRESSION [VALUE=LOG(P1/(1-P1))] Transformation
CALCULATE Y = #Transformation
```

If you are writing a procedure that takes an expression as one of its inputs, you may want to know what results it is generating and what data structures it is using to calculate them. The FARGUMENTS allows you to find this out.

The expression to study is specified by the EXPRESSION option. The NRESULTS option can save the number of results, and the NCALCULATIONS option can save the number of calculations. The parameters of FARGUMENTS allow you to save information about each of the calculations in the expression: the ICALCULATION parameter specifies the number of the calculation, the RESULT parameter can specify a dummy to be set to the structure that is given the result, and the ARGUMENTS parameter can specify a pointer to save the arguments.

Options: EXPRESSION, NRESULTS, NCALCULATIONS.
Parameters: ICALCULATION, RESULT, ARGUMENTS.

See also

Directives: EXPRESSION, DUMMY, POINTER, SCALAR, FCLASSIFICATION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

FAULT

Checks whether to issue a diagnostic, i.e. a fault, warning or message.

Options

DIAGNOSTIC = *string token*	Severity of the diagnostic (`fault`, `warning`, `message`); default `faul`
FAULT = *text*	Diagnostic code; default `'UF 1'` for fault, `'UF 2'` for warning
EXPLANATION = *text*	Explanatory information

Parameter

expression	Logical expression to test whether or not to give the diagnostic

Description

FAULT can be used to generate a Genstat fault, warning or message, as requested by the DIAGNOSTIC option. The diagnostic is printed in the standard Genstat format. So, for example, faults and warnings are recognised by Genstat *for Windows*, and added to the Event Log. Also, the diagnostic will be suppressed (like those from Genstat directives) if that has been requested by the DIAGNOSTICS option of the SET directive.

There is a single parameter, which supplies a logical expression to decide whether or not to give the diagnostic; if this is omitted, the diagnostic is always given. The FAULT option defines the code to identify a fault or warning; this has a default of `'UF 1'` for a fault and `'UF 2'` for a warning. (Messages always begin with the standard prefix "Message: ".) The EXPLANATION option allows you to supply some explanatory information.

FAULT is particularly useful in procedures. For example, in a regression procedure, you might put

```
FAULT [DIAGNOSTIC=fault; FAULT='VA 6';\
  EXPLANATION='Y-variate must contain at least 2 values']\
  NOBSERVATIONS(Y) < 2
```

Then, if the y-variate has less than two non-missing values, Genstat will give a "VA 6" fault, and execution of the procedure will stop. The available Genstat faults are listed in the on-line help.

Options: DIAGNOSTIC, FAULT, EXPLANATION.
Parameter: unnamed.

See also

Directives: DISPLAY, EXIT, PROCEDURE, CALCULATE.
Genstat Reference Manual 1 Summary section on: Program control.

FCA

Performs factor analysis.

Options

PRINT = *string tokens*	Printed output required (communalities, loadings, coefficients, scores, residuals, cresiduals, vresiduals, tests); default * i.e. no printing
NDIMENSIONS = *scalar*	Number of factors to fit; no default, must be specified
METHOD = *string token*	Whether to use correlations or variances and covariances (correlation, vcovariance, variancecovariance); default vcov
MAXCYCLE = *scalar*	Maximum number of iterations; default 50
TOLERANCE = *scalar*	Minimum value to assume for the unique component ψ_i^2 of each observed variable; default 10^{-6}

Parameters

DATA = *pointers* or *matrices* or *symmetric matrices* or *SSPMs*	Pointer of variates forming the data matrix, or matrix storing the variate values by columns, or symmetric matrix storing their variances and covariances, or SSPM giving their sums of squares and products
NUNITS = *scalars*	When DATA is set to a symmetric matrix of variances and covariances, NUNITS must specify the number of units from which they were calculated if tests are required
LRV = *LRVs*	Saves the loadings, latent roots and trace from each analysis
SSPM = *SSPMs*	Saves the SSPM formed from a DATA matrix or pointer
COMMUNALITIES = *variates*	Saves the communalities
COEFFICIENTS = *matrices*	Saves the factor score coefficients
SCORES = *matrices* or *pointers*	Saves the factor analysis scores
RESIDUALS = *matrices* or *pointers*	Saves residuals from the dimensions fitted in the analysis
CRESIDUALS = *symmetric matrices*	Saves the residual correlation or covariance matrix
VRESIDUALS = *variates*	Saves the residual variances

Description

Factor analysis aims to find a set of "latent" (or unobservable) variables $\{z_1...z_k\}$ that account for the variances and covariances **S** between a set of p observed variables $\{x_1...x_p\}$. In the terminology of factor analysis, the latent variables $\{z_i\}$ are known as *factors*. However, they are continuous variables, and thus are represented in Genstat by variate rather than by factor data structures. So to avoid confusion, when we refer to the latent variables below, *factor* will be printed in italic font.

The data for a factor analysis consists of observed measurements on the variables $\{x_i\}$ made on a set of subjects. The assumption is that, for each subject, the values of the observed variables are related to the *factors* by a linear model

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\Gamma}\,\boldsymbol{z} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{x}$ is the vector of observed variables,

$\boldsymbol{z}$ is the vector of *factors*,

$\boldsymbol{\mu}$ is a vector of means for the observed variables,

$\boldsymbol{\Gamma}$ is a matrix of *loadings* defining the relationship between observed and latent variables,

and

$\boldsymbol{\varepsilon}$ is a vector of residuals.

The elements of the residual vector $\boldsymbol{\varepsilon}$ are assumed to have mean zero and to be uncorrelated, i.e. the dispersion matrix of $\boldsymbol{\varepsilon}$ is assumed to be diagonal

$$\mathrm{cov}(\boldsymbol{\varepsilon}) = \boldsymbol{\Psi} = \mathrm{diag}(\psi_1^2, \ldots \psi_p^2)$$

(They thus differ from the residuals formed in a principal components analysis, which will be correlated; see e.g. Krzanowski 1988 Section 16.2 for more details). The *factors* themselves are assumed to have variance one and to be uncorrelated, i.e.

$$\mathrm{cov}(\boldsymbol{z}) = \mathbf{I}.$$

So the correlations between the observed variables $\{x_i\}$ arise only through their relations with the *factors*, and not because of any correlation between the residuals or between the *factors*.

The `DATA` parameter specifies the data for the factor analysis. You can supply either a pointer containing a set of variates, one for each observed variable $\{x_i\}$, or a matrix storing the observed variables by columns, or a symmetric matrix containing variances and covariances between the variables, or an SSPM structure (formed using `FSSPM` from the variates of observed measurements). When `DATA` specifies a symmetric matrix of variances and covariances, you must also set the `NUNITS` parameter to specify the number of units from which they were calculated if you want `FCA` to print tests.

The `METHOD` option has settings `vcovariance` (with synonym `variancecovariance`) and `correlation`, to control whether `FCA` forms a matrix of variances and covariances or a matrix of correlations for the analysis. The same *factors* will be obtained if you use a correlation matrix, but the loadings will be scaled to be between zero and one. The number of *factors*, q, to fit must be specified by the `NDIMENSIONS` option. Arising from the numbers of parameters in the model (see Krzanowski 1988 Section 16.2.2) this is subject to the constraint

$$(p - q)^2 \geq p + q.$$

The `PRINT` option controls printed output, with settings:

`communalities`	the proportion of variation explained by the *factors* for each observed variable, $(\mathrm{var}(x_i) - \psi_i^2) / \mathrm{var}(x_i)$;
`loadings`	the matrix of factor loadings $\boldsymbol{\Gamma}$;
`coefficients`	the factor score coefficients;
`scores`	the factor scores calculated from the model for each subject;
`residuals`	the vectors of residuals ε,
`cresiduals`	the residual correlation or covariance matrix i.e. a symmetric matrix showing the amount of unexplained correlation or covariance between each pair of variables;
`vresiduals`	the residual variances; and
`tests`	a chi-square goodness of fit test for the model.

By default nothing is printed. Note, however, that scores and residuals cannot be produced when `DATA` is set to a symmetric matrix of variances and covariances.

The communalities, factor coefficients, scores, residuals, residual correlations or covariances and residual variances can also be saved using the `COMMUNALITIES`, `COEFFICIENTS`, `SCORES`, `RESIDUALS`, `CRESIDUALS` and `VRESIDUALS` parameters, respectively. The `LRV` parameter allows an LRV structure to be saved, with the loadings in the `['vectors']` component, and the eigenvalues of the matrix $\boldsymbol{\Psi}^{-½}\,\mathbf{S}\,\boldsymbol{\Psi}^{-½}$ in the `['roots']` component; the loadings are scaled eigenvectors of $\boldsymbol{\Psi}^{-½}\,\mathbf{S}\,\boldsymbol{\Psi}^{-½}$. (Remember, $\mathbf{S}$ is the matrix of variances and covariances of the observed variables $\{x_i\}$.) The `SSPM` parameter can save the SSPM structure constructed from a `DATA` pointer for the analysis. A particularly convenient instance is when you have supplied an SSPM structure as input but, for example, have set `METHOD=correlation`: the SSPM that is saved will then contain correlations instead of sums of squares and products.

Options: PRINT, NDIMENSIONS, METHOD, MAXCYCLE, TOLERANCE.
Parameters: DATA, NUNITS, LRV, SSPM, COMMUNALITIES, COEFFICIENTS, SCORES, RESIDUALS, CRESIDUALS, VRESIDUALS.

Method
FCA estimates the parameters of the model by maximum likelihood, assuming multivariate Normality, using subroutines G03CAF and G03CCF from the NAG Library. The MAXCYCLE option sets a limit on the number of iterations (default 50). The TOLERANCE option specifies the minimum value to assume for the unique component ψ_i^2 of each observed variable so that the communality is always less than one; the default is 10^{-6}.

Action with RESTRICT
If any of the variates in a DATA pointer is restricted, only the defined subset of the units will be used in the analysis.

References
Krzanowski, W.J. (1988). *Principles of Multivariate Analysis: a User's Perspective*. Oxford University Press, Oxford.

See also
Directives: CVA, MDS, PCO, PCP, ROTATE, SSPM.
Procedures: LRVSCREE, DMST, PLS, RIDGE.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

FCLASSIFICATION

Forms a classification set for each term in a formula, breaks a formula up into separate formulae (one for each term), and applies a limit to the number of factors and variates in the terms of a formula.

Options

FACTORIAL = *scalar*	Limit on the number of factors and variates in each term; default * i.e. no limit
NTERMS = *scalar*	Outputs the number of terms in the formula
CLASSIFICATION = *pointer*	Saves a list of all the factors and variates in the TERMS formula
OUTFORMULA = *formula structure*	Identifier of a formula to store a new formula, omitting terms with too many factors and variates
INCLUDEFUNCTIONS = *string token*	Whether to include functions in the formulae saved by the OUTFORMULA option or the OUTTERMS parameter (yes, no); default no
REORDER = *string token*	When to reorder the terms in the model (always, standard, never); default stan
DROPTERMS = *string token*	Whether to include only terms that can be dropped individually from the formula (yes, no); default no
CHECKFUNCTIONS = *scalar*	Indicator, set to one if the TERMS formula contains any functions, and zero if it contains none
FUNCTIONDEFINITIONS = *pointer*	Saves details of the functions defined for each factor and variate in the TERMS formula
EXCLUDEPSEUDOTERMS = *string token*	Whether to omit pseudo-terms from the number of terms and the formulae saved by the OUTFORMULA option and OUTTERMS parameter (yes, no); default no

Parameters

TERMS = *formula*	Formula from which the classification sets, individual model terms and so on are to be formed
CLASSIFICATION = *pointers*	Identifiers for pointers to store the factors and variates composing each model term of the TERMS formula
OUTTERMS = *formula structures*	Identifiers for formulae to store each individual term of the TERMS formula
MAINTERMS = *formula structures*	Identifiers for formulae to store the main term for each individual term of the TERMS formula

Description

If you are writing procedures, for example for statistical analyses, the model to be fitted will often be specified by a Genstat formula structure. Unless the algorithm within the procedure merely involves straightforward use of one of Genstat's statistical directives, you may wish to know more about the formula: how many model terms does it contain, which factors do they involve, and so on. The FCLASSIFICATION directive is designed to provide the answers to these questions. The formula is specified using the TERMS parameter.

When Genstat uses a formula in a statistical analysis, it is expanded into a series of model terms, linked by the operator +. FCLASSIFICATION allows you to save this expanded form, in another formula, using the OUTFORMULA option.

You can use the FACTORIAL option to apply a limit to the number of factors and variates in

the resulting terms, similarly to the FACTORIAL option in the ANOVA, FIT or REML directives. The number of terms in the formula can be saved (in a scalar) using the NTERMS option, and a list of the factors and variates that occur in the formula can be saved (in a pointer) using the CLASSIFICATION option.

The other parameters allow you to save information about the individual model terms in the formula. The identifiers in the lists that they specify are taken in parallel with the model terms in the expanded form of the formula. For each model term, the corresponding identifier in the list for the CLASSIFICATION parameter is defined as a pointer storing the factors that occur in the term. The identifier in the OUTTERMS list is defined as a formula containing just that model term.

The MAINTERMS parameter is useful if the formula contains pseudo-factors. Its identifiers save formula structures containing the "main term" for each of the model terms. If the term is a pseudo-term, this will be the model term to which the pseudo-term is linked. Otherwise, it will be the term itself. For example, in the model

```
Variety//(A+B)
```

in Example 4.7.3c in the *Guide to Genstat, Part 2 Statistics*, there are two pseudo-terms, A and B, with Variety as their main term.

You can set option EXCLUDEPSEUDOTERMS = yes to omit pseudo-terms from the saved information (number of terms, OUTFORMULA or OUTTERMS).

By default any functions such as POL or REG are omitted from the formulae saved by OUTFORMULA or OUTTERMS, but these will be included if you set option INCLUDEFUNCTIONS=yes. The CHECKFUNCTIONS option allows you to save a scalar containing one if the TERMS formula contains any functions, and zero if it does not.

The FUNCTIONDEFINITIONS option allows you to obtain details of the functions. This saves a pointer which contains a pointer for each factor and variate in the formula (in the same order as in the CLASSIFICATION pointer). If the factor or variate has no function, its pointer contains just a text with a single missing value (''). Otherwise the first element of the pointer is a text containing the name of the function (either 'POL', 'POLND', 'REG', 'REGND', 'COMP', 'SSPLINE' or 'LOESS'). It then contains elements to store the second and subsequent arguments of the function (if any).

Model terms involving several factors are regarded by Genstat as representing all the joint effects of these factors that are not removed by earlier terms in the formula. So, in the formula

```
A + B + A.B
```

A.B is the interaction of factors A and B, as both main effects occur earlier in the formula. Alternatively, in the formula

```
A.B + A + B
```

A.B still represents all the joint effects of factors A and B, and the later terms A and B are redundant as they are now "contained" in A.B. Thus FCLASSIFICATION usually deletes any term in the model that is contained in an earlier term. However, if you set option REORDER=always, the model is reordered after applying any operator (including plus). The reordering arranges the terms so that they contain increasing numbers of identifiers. Terms with the same number of identifiers are then put into lexicographical order with respect to the order in which the identifiers first occurred in the formula itself. Each term will therefore come before any term that would contain it. So the model would again be

```
A + B + A.B
```

The default setting, REORDER=standard, applies the standard Genstat rules, which reorder the terms only after a dot, slash or star operator. The final setting REORDER=never specifies that no reordering should take place. (Before Release 19.2, the ORTHOGONAL option had settings no and yes, corresponding to standard and always. Options and parameters with settings yes and no should not have any other settings. So these were renamed in Release 19.2, when the

setting `never` was added. However, `no` and `yes` are retained as synonyms, so that earlier programs will still run.)

The rules about terms that contain other terms are also relevant when you are dropping terms from a model, for example in a regression analysis. You cannot drop a term, for example using the `DROP` directive, until all the terms that contain it have been dropped. To simplify the process, if you set option `DROPTERMS=yes`, the formulae saved by `OUTFORMULA` or `OUTTERMS` will contain only terms that are not contained in any other terms (i.e. only the terms that can be dropped).

Options: `FACTORIAL`, `NTERMS`, `CLASSIFICATION`, `OUTFORMULA`, `INCLUDEFUNCTIONS`, `REORDER`, `DROPTERMS`, `CHECKFUNCTIONS`, `FUNCTIONDEFINITIONS`, `EXCLUDEPSEUDOTERMS`.
Parameters: `TERMS`, `CLASSIFICATION`, `OUTTERMS`, `MAINTERMS`.

See also
Directives: `FORMULA`, `FARGUMENTS`, `REFORMULATE`, `SETCALCULATE`, `SETRELATE`, `SET2FORMULA`.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

FCOPY

Makes copies of files.

No options

Parameters

OLD = *texts*	Name of each file to copy
NEW = *texts*	Name for the new copy of each file
OVERWRITE = *string tokens*	Whether to overwrite any existing files (yes, no); default no

Description

FCOPY allows you to make a copy of an external file. The names of the original files are specified, in texts, but the OLD parameter. The file names for the copies are specified by the NEW parameter. If no path is included in the file name, it is assumed to be in the current working directory (which can be defined by the WORKINGDIRECTORY option of the SET directory). If you need to define the path, remember that the character \ is the continuation symbol in Genstat. So this character needs to be duplicated in a string to avoid Genstat interpreting it as a continuation: for example

```
FCOPY 'Today.Dat'; NEW='D:\\April\\18.dat'
```

copies the file Today.dat to the file 18.dat in the directory (or folder) D:\April. As a more convenient alternative, the PC version of Genstat allows you to use / instead: i.e. you could put

```
FCOPY 'Today.Dat'; NEW='D:/April/18.dat'
```

By default FCOPY gives a fault if the new file already exists, but you can set parameter OVERWRITE=yes to overwrite it.

Options: none.
Parameters: OLD, NEW, OVERWRITE.

See also

Directives: FDELETE, FRENAME, CLOSE, ENQUIRE, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

FCOVARIOGRAM

Forms a covariogram structure containing auto-variograms of individual variates and cross-variograms for pairs from a list of variates.

Options

PRINT = *string token*	Controls printed output (statistics, variograms, autovariograms); default stat
METHOD = *string token*	Specifies what to do when the measurements are not all made at the same locations (allwithcrossnugget, allnocrossnugget, commonpoints); default comm
COVARIOGRAM = *pointer*	Pointer to store the variograms, cross-variograms and associated information for use in MCOVARIOGRAM
MAXLAG = *scalar*	Maximum lag in all directions
STEPLENGTHS = *scalar* or *variate*	Length of the step or steps in which lag is incremented
DIRECTIONS = *scalar* or *variates*	Directions along which to form the variogram, scalar for a single direction in 2 dimensions, variate for several directions in 2 dimensions, and pairs of variates for 3 dimensional data
SEGMENTS = *scalar*	Angle subtended by each segment along the DIRECTIONS
COORDSYSTEM = *string token*	Coordinate system used for the geometry for discretizing the lag (mathematical, geographical); default math
MAXCONEDIAMETER = *scalar*	Diameter at which the segments over which averaging is to be done should cease to expand; default * implies no limit
MINCOUNT = *scalar*	Minimum number of points required at a particular lag point for the cross-variogram to be estimated there; default 1
DRIFT = *string token*	Mean function (constant, linear, quadratic); default cons

Parameters

DATA = *variates*	Measurements as a variate
X1 = *variates*	Locations of each set of measurements in the first dimension
X2 = *variates*	Locations of each set of measurements in the second dimension (if recorded in more than 1 dimension)
X3 = *variates*	Locations of each set of measurements in the third dimension (if recorded in 3 dimensions)

Description

The FCOVARIOGRAM directive forms a covariogram structure containing auto- and cross-variograms for pairs from a list of variates.

The data are supplied as a list of variates using the DATA parameter, where each variate contains the measurements for each variable. The locations of the measurements are supplied using parameters X1, X2 (for two or three dimensions) and X3 (for three dimensions) parameters.

The METHOD option specifies how to calculate the cross-variograms. The setting commonpoints specifies that only those points observed in common in every sample are to be included; the method described in Section 8.3.4 of Part 2 of the *Guide to the Genstat Command Language* are then used. Alternatively, the setting allnocrossnugget can be used when the sampling locations do not match. This uses an algorithm outlined in Künsch, Papritz & Bassi

(1997) that performs least-squares fitting of the cloud of products of differences to estimate the expected value of these products. If there are no common points, the nugget variance cannot be calculated. However, if there is partial sampling (i.e. some common points), the setting `allwithcrossnugget` can be used to shift the cross-variograms by the semivariance at the origin to estimate the nugget effect.

The maximum lag distance in all directions to which the variograms are calculated is set by the `MAXLAG` option. The increments in distance are set by the `STEPLENGTH` option, where you can supply a scalar to define equally-spaced steps or a variate to specify the steps themselves. The directions along which to form the variograms are supplied in degrees using the `DIRECTIONS` option. The geometry used for the directions is given by the `COORDSYSTEM` option: the setting `mathematical` specifies directions counter-clockwise from east, and `geographical` specifies clockwise from north (for the first direction only in three dimensions). Each direction is at the centre of an angular range. The angle is the same in every direction, and is defined by the `SEGMENTS` option. For a single direction in two dimensions the `DIRECTIONS` option should be set to a scalar, while for several directions it should be set to a variate. For directions in three dimensions, `DIRECTIONS` should specify a pair of variates. The `MAXCONEDIATMETER` option can be used to specify a diameter at which the segments cease to expand. For cross-variograms that are formed using all points the minimum number of points required at each lag can be specified using the `MINCOUNT` option.

The `DRIFT` option can be used to calculate the variograms after removing a systematic component. Setting the `DRIFT` option to linear or quadratic will fit a regression to the observations and then form the variograms on the residuals.

The `COVARIOGRAM` option allows you to specify pointer to save the auto-variograms, cross-variograms and associated information. Its elements contain:

1 a matrix with columns of variograms and cross-variograms and rows indexed by lags within directions;
2 a variate of counts at the lags in each direction;
3 distances of the lags in each direction;
4 horizontal angles;
5 vertical angles;
6 variances;
7 distance classes;
8 method;
9 pointer containing identifiers of the `DATA` variates;
10 number of dimensions.

This structure provides the information required to fit models to the covariogram using the directive `MCOVARIOGRAM`.

The `PRINT` option can be set to statistics to display statistics for each of the variates. The setting `variograms` displays each of the auto- and cross-variograms, while the setting `autovariogram` displays only the auto-variograms.

Options: `PRINT`, `METHOD`, `COVARIOGRAM`, `MAXLAG`, `STEPLENGTHS`, `DIRECTIONS`, `SEGMENTS`, `COORDSYSTEM`, `MAXCONEDIAMETER`, `MINCOUNT`, `DRIFT`.
Parameters: `DATA`, `X1`, `X2`, `X3`.

Action with `RESTRICT`
Restrictions are ignored.

Reference
Künsch, H.R., Papritz, A. & Bassi, F. (1997) Generalized cross-covariances and their estimation. *Mathematical Geology*, **29**, 779-799.

See also
Directives: MCOVARIOGRAM, COKRIGE, FVARIOGRAM, KRIGE.
Procedures: DCOVARIOGRAM, KCROSSVALIDATION, MVARIOGRAM, DVARIOGRAM, DHSCATTERGRAM.
Genstat Reference Manual 1 Summary section on: Spatial statistics.

FDELETE

Deletes files.

No options

Parameter

NAME = *texts*	Names of the files to delete

Description

FDELETE allows you to delete external file. The names of the files are specified, in texts. These can include the wildcards * (to mean any sequence of characters) and ? (to mean any single character). If no path is included in the file name, it is assumed to be in the current working directory (which can be defined by the WORKINGDIRECTORY option of the SET directory). If you need to define the path, remember that the character \ is the continuation symbol in Genstat. So this character needs to be duplicated in a string to avoid Genstat interpreting it as a continuation: for example

```
FDELETE 'D:\\Data\\Tempfile.dat'
```

deletes the file Tempfile.dat in the directory (or folder) D:\Data. As a more convenient alternative, the PC version of Genstat allows you to use / instead: i.e. you could put

```
FDELETE 'D:/Data/Tempfile.dat'
```

Options: none.
Parameter: NAME.

See also

Directives: FCOPY, FRENAME, CLOSE, ENQUIRE, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

FILTER

Filters time series by time-series models (synonym of TFILTER).

Option

PRINT = *string tokens*	What to print (series); default *

Parameters

OLDSERIES = *variates*	Time series to be filtered
NEWSERIES = *variates*	To save filtered series
FILTER = *TSMs*	Models to filter with respect to
ARIMA = *TSMs*	ARIMA models for time series

Description

FILTER was renamed as TFILTER in Release 14 to emphasize its status as a time-series command. The earlier name (FILTER) was retained to allow previous programs to continue to run, but this may be removed in a future release.

Filtering is a means of processing a time series so as to produce a new series. The purpose is usually to reveal some features and remove other features of the original series. Filters in Genstat are one-sided: that is, each value in the new series depends only on present and past values of the original series. However, you can do two-sided filtering by using the SHIFT and REVERSE functions of CALCULATE.

The OLDSERIES and NEWSERIES parameters of FILTER specify respectively the time series to be filtered, and the series that result from filtering. A new series must not have the same identifier as the series from which it was calculated. Genstat interprets any missing values in the old series as zero. But if you use the ARIMA parameter (see below), Genstat replaces them by interpolated values when it calculates the filtered series; the missing values remain in the old series.

The FILTER parameter specifies the TSMs to be used for filtering. If the TSM is a transfer-function model, the new series y_t is calculated from the old series x_t by

$$y_t = \{ \omega(B)B^b / \delta(B)\nabla^d \} x_t.$$

The filter does not use the power transformation nor the reference constant. This lets you apply a single filter conveniently to a set of time series, for which different transformations and different constants might be appropriate. You can always use the CALCULATE directive to apply a transformation to a series before using FILTER.

If the TSM is an ARIMA model, then the new series a_t is calculated from the old series y_t by

$$a_t = \{ \varphi(B)\nabla^d / \theta(B)\} y_t.$$

Note that the TSM does not have to be the model appropriate for y_t. Again, Genstat ignores the parameters λ, c and σ_a^2; you can set them to 1,0,0, for example.

The ARIMA parameter specifies a time-series model for the old series. The purpose is to reduce transient errors that arise in the early part of the new series: these arise because Genstat does not know the values of the old series that came before those that have been supplied. If you do not use this parameter, then Genstat takes these earlier values to be zero. This can cause unacceptable transients which can only be partially removed by procedures such as mean-correcting the old series. If you do use the ARIMA parameter, then Genstat uses the specified model to estimate (or back-forecast) the values of the old series earlier than those that have been supplied.

You do not have to have a good ARIMA model for the old series in order to achieve worthwhile reductions in the transients. Thus a model with orders (0,1,1) and parameters (1,0,0,0.7) would estimate the prior values to be constant, at a level that is a backward EWMA of the early values of the series.

For a seasonal monthly time series, an appropriate ARIMA model could have orders

(0,1,1,0,1,1,12) and parameters (1,0,0,0.7,0.7). However you must give the supplied model a transformation parameter λ=1. Any other value for λ breaks the assumption of linearity that underlies the calculations for correcting the transients. The constant term in the ARIMA model can be non-zero, and should be if that is appropriate for the old series. Note that the ARIMA model does not define the filter.

If you specify the ARIMA parameter, Genstat uses this model to interpolate any missing values in the old series before it calculates the new series. Suppose for example that the filter is the identity, defined by a transfer-function model with orders (0,0,0,0) and parameters (1,0,0); then the new series will be the old series with any missing values replaced.

Option: PRINT.
Parameters: OLDSERIES, NEWSERIES, FILTER, ARIMA.

Action with RESTRICT
The OLDSERIES variate can be restricted, but this must be to a contiguous set of units.

See also
Directive: TFILTER.
Genstat Reference Manual 1 Summary section on: Time series.

FIT

Fits a linear, generalized linear, generalized additive or generalized nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, grid, confidence); default mode, summ, esti or grid if NGRIDLINES is set
CALCULATION = *expression structures*	Calculation of explanatory variates involving nonlinear parameters
OWN = *scalar*	Option setting for OWN directive if this is to be used rather than CALCULATE to calculate explanatory variates
CONSTANT = *string token*	How to treat the constant (estimate, omit, ignore); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default as in previous TERMS statement, or 3 if no TERMS given
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
NGRIDLINES = *scalar*	Number of values of each nonlinear parameter for a grid of function evaluations
SELINEAR = *string token*	Whether to calculate s.e.s for linear parameters when nonlinear parameters are also estimated (yes, no); default no
INOWN = *identifiers*	Setting to be used for the IN parameter of OWN if used to calculate explanatory variates
OUTOWN = *identifiers*	Setting to be used for the OUT parameter of OWN if used to calculate explanatory variates
AOVDESCRIPTION = *text*	Description for line in accumulated analysis of variance

(or deviance) table when POOL=yes

Parameter

formula	List of explanatory variates and factors, or model formula

Description

A FIT statement must always be preceded by a MODEL statement, though not necessarily immediately. You can give several FIT statements after a single MODEL statement; for example, to try out different explanatory variables.

The parameter of the FIT directive specifies the explanatory variables in the model. In simple regression, it consists of the identifier of a single explanatory variate. If you omit the parameter, Genstat fits a *null model*; that is, a model consisting of just one parameter, the overall mean. In multiple regression the parameter consists of a list of explanatory variates, and factors may also appear to include the main effects of qualitative explanatory variables.

More generally, the parameter may be in the form of a model formula, including interactions between explanatory variables and functions of explanatory variables. The interaction between two or more variates is interpreted as another variate formed from the product of the constituent variates. The interaction between factors is interpreted as in the TREATMENTSTRUCTURE directive; and in general the expansion of model formulae is controlled by the FACTORIAL option in the same way as in the ANOVA directive. The interaction between a variate and a factor represents differential responses for the variate at each level of the factor, and similarly if several variates or factors are involved. A formula may also include POL, REG and COMPARISON functions of variates or factors, representing polynomial contrasts (up to order 4), orthogonalized regression or polynomial contrasts (up to order 8) and non-orthogonalized regression contrasts (up to order 8) respectively. Variates may also appear in SSPLINE functions, representing cubic smoothing spline effects with specified numbers of degrees of freedom or specified smoothing parameters. Similarly, variates may appear in LOESS functions, representing smoothed effects from locally weighted regressions. Multi-dimensional smoothing can be achieved by supplying a pointer containing up to four variates as the first argument of LOESS. Models including such terms are called *additive* or *generalized additive models* (Hastie & Tibshirani 1990). Smoothed variates may also appear in interactions, where they represent the same effects as if the variate did not appear in the SSPLINE function; the model then fits a common smooth effect in addition to the usual linear effects for each combination of factor levels.

The CALCULATION option allows you to specify one or more expressions to be evaluated before carrying out the linear or generalized linear fit. This is only done if an RCYCLE statement has been given to list nonlinear parameters. The expressions can then make use of the current values of the nonlinear parameters to derive components of the fitted model. At each stage of the nonlinear search for the best estimates of these parameters, the linear or generalized linear model is fitted after evaluating the expressions with the current values of these parameters. Models of this kind are referred to as *generalized nonlinear models* (Lane 1996).

The PRINT option controls output. You can give several settings at the same time, to provide reports on several aspects of the analysis. The model setting gives a description of the model, including response and explanatory variates.

The output from the summary setting is a summary analysis of variance, or analysis of deviance in generalized linear models. The summary includes F-probabilities if option FPROBABILITY=yes, but the interpretation of these probabilities depends on the usual assumptions of regression analysis, and they are only approximate in generalized linear models. Following the analysis of variance further information is presented about the fit of the model, the contents of which are controlled by the SELECTION option. By default, for models with the Normal distribution, this consists of the percentage variance accounted for and the standard error

of the observations. The Percentage variance accounted for is the *adjusted R^2 statistic*, expressed as a percentage: 100 × (1 − (Residual m.s.)/(Total m.s.)). The standard error of the observations is estimated by the square root of the residual mean square. For the gamma distribution, the default is to display the coefficient of variation instead, while for other distributions the default is to display the dispersion. The setting `aic` presents the Akaike information criterion, and the settings `bic` and `sic` are synonyms that present the Schwarz (Bayesian) information criterion (see Koehler & Murphree 1988 for a comparison); the values calculated by Genstat omit some constant terms that depend on the data rather than the model, so it is the differences between values for different models that should be of interest rather than the absolute values. There may also be messages in the output, produced as a result of several checks made by Genstat on the adequacy of the model. Extreme residuals and leverage values are reported, and simple checks are made on constancy of variance and systematic departure from the fitted model. You can prevent these messages appearing by using the `NOMESSAGE` option. They will not appear in any case if you have set option `RMETHOD=*` in the `MODEL` statement.

The `estimates` setting produces the estimates of parameters in the model. The standard errors of the estimates are based by default on the residual mean square. Alternatively, you can supply an estimate of variance by using the `DISPERSION` option of `MODEL`; if you do this, Genstat will print a reminder about the basis of the standard errors. You can prevent this reminder appearing by setting the `NOMESSAGE` option. T-statistics are also displayed, allowing you to test whether each parameter differs significantly from zero, keeping the other parameters fixed; these probabilities too depend on the usual assumptions of regression analysis. The number of degrees of freedom for such a test appears in the column heading. If the estimate of variance is supplied, then the "t-statistics" actually have a standard Normal distribution, indicated by the column heading "t(*)". If the `TPROBABILITY` option is set, the corresponding probabilities are displayed. You can also display confidence intervals for the parameters by including the `confidence` setting. The probability value for the intervals is set by the `PROBABILITY` option; default 0.95.

The variance inflation factor is calculated for each parameter, and a message is generated if any is greater than 100, to warn that some explanatory terms are nearly aliased and that the standard errors of their parameters are consequently inflated. The parameters involved in the relationship are listed with the inflation factors. The variance inflation factor is defined to be the current diagonal value of the inverse matrix $(X^TX)^{-1}$ corresponding to the parameter, multiplied by the corrected sum of squares of the variate or dummy variate corresponding to the parameter. This can be interpreted as the ratio of the variance of the parameter estimate in the current model compared with that of the estimate in a model containing just that parameter and the constant. The check will not be made if the current model contains any `POL` submodels, or any term involving interaction between a variate and a factor, because the dummy variates generated to represent these effects are very likely to be nearly aliased with each other. The check is also omitted if the constant term is excluded from the model. When a generalized linear model is fitted with a log or logit link function, the antilogs of the parameters are also displayed, to summarize their multiplicative effects on the natural or odds scale respectively.

For a linear model with Normally distributed response, the `accumulated` setting displays an analysis attributing the variance of the explanatory terms in the order in which they are given in the parameter of `FIT`; no subdivision is available for generalized linear or nonlinear models unless terms are explicitly added or dropped one at a time using further directives such as `ADD`, `DROP` or `SWITCH`. The subdivision is also not made if the `POOL` option is set to `yes`. The denominator of the ratios in the analysis can be controlled by setting the `DENOMINATOR` option. The lines of the accumulated table are usually labelled by the names of the model terms that have been added or dropped. When `POOL=yes`, however, this may become rather too long or complicated, so you can then use the `AOVDESCRIPTION` option to supply your own description. If you supply a null text (containing just a single, empty line), the line is omitted from the table.

The `deviance` setting produces an abbreviated summary of the analysis. The `correlations` setting gives a correlation matrix of the parameter estimates. The `fitted` setting displays a table of unit labels, values of response variate, fitted values, standardized residuals and leverages. The `monitoring` setting reports the progress of any iterative search, as used in generalized linear, additive and nonlinear models. Finally, the `grid` setting is relevant only for generalized nonlinear models when the `NGRIDLINES` option is set, as in `FITNONLINEAR`.

The `CONSTANT` option controls whether the constant parameter is included in the model. In simple linear regression, this parameter is the intercept, in other words the estimate of the response variable when the explanatory variable is zero. In models containing factors, the constant will be the parameter corresponding to the reference level of the factor or factors, and the estimates printed for other levels will be differences between the parameter for those levels and that for the reference level (for more details, see the *Guide to the Genstat Command Language*, Part 2, Section 3.3.2). Consequently, the constant should then not be omitted unless the `FULL` option of `TERMS` has been set to ensure that the model contains a parameter for every level of the factor. If you set `CONSTANT=omit` for a model containing factors without setting `FULL=yes` in `TERMS`, Genstat gives a failure diagnostic. The diagnostic can be suppressed by setting `CONSTANT=ignore` instead, but this should be done only in special circumstances (as, for example, inside the procedure `HGANALYSE` which fits hierarchical generalized linear models).

The `NOMESSAGE` option controls printing of messages. The `aliasing` setting suppresses messages about aliasing of parameters, and the `marginality` setting suppresses reports of violation of marginality principles when fitting interactions between explanatory variables. The `leverage` setting prevents messages about large leverages, and `residual` prevents messages about large residuals or non-constant variance or systematic pattern in the residuals. The `inflation` setting suppresses messages about the variance inflation factor, and the `dispersion` setting prevents reminders appearing about the basis of the standard errors (as can be produced by the `estimates` setting of the `PRINT` option).

The `OWN`, `INOWN` and `OUTOWN` options are as in the `FITNONLINEAR` directive, and allow the model calculations for a generalized nonlinear model to be specified in a lower-level language, such as Fortran. The `NGRIDLINES` and `SELINEAR` options are also relevant to these models only, and provide a grid of functions values and standard errors of linear parameters, respectively, as in `FITNONLINEAR`.

After fitting a regression using `FIT`, the model can be modified using the `ADD`, `DROP`, `STEP`, `SWITCH` and `TRY` directives, further output can be displayed using the `RDISPLAY` directive, and results can be copied into Genstat data structures using the `RKEEP` directive. The fit can be assessed graphically using the procedures `RGRAPH` and `RCHECK`.

Options: `PRINT`, `CALCULATION`, `OWN`, `CONSTANT`, `FACTORIAL`, `POOL`, `DENOMINATOR`, `NOMESSAGE`, `FPROBABILITY`, `TPROBABILITY`, `SELECTION`, `PROBABILITY`, `NGRIDLINES`, `SELINEAR`, `INOWN`, `OUTOWN`, `AOVDESCRIPTION`.
Parameter: unnamed.

Action with `RESTRICT`

You can restrict the units that Genstat will use for the regression by putting a restriction on any of the vectors involved in the `MODEL` statement (response variates, weight variate, offset variate, grouping factor or variate of binomial totals), or on any explanatory variate or factor. However, you are not allowed to have different restrictions on the different vectors.

References

Hastie, T.J. & Tibshirani, R.J. (1990). *Generalized Additive Models*. Chapman and Hall, London.

Koehler, A.B. & Murphree, E.S. (1988). A comparison of the Akaike and Schwarz criteria for selecting model order. *Applied Statistics*, **37**, 187-195.

Lane, P.W. (1996). Generalized nonlinear models. *COMPSTAT 1996 Proceedings in Computational Statistics* (ed. Prat, A.), 331-336.

See also

Directives: MODEL, TERMS, RDISPLAY, PREDICT, RKEEP, RKESTIMATES, ADD, DROP, SWITCH, STEP, TRY, FITCURVE, FITNONLINEAR, RCYCLE, RFUNCTION.

Procedures: RCHECK, RGRAPH, RPERMTEST, RWALD, FITINDIVIDUALLY, FITMULTINOMIAL, GLMM, HGANALYSE, RAR1.

Functions: COMPARISON, POL, REG, LOESS, SSPLINE.

Genstat Reference Manual 1 Summary section on: Regression analysis.

FITCURVE

Fits a standard nonlinear regression model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring); default mode, summ, esti
CURVE = *string token*	Type of curve (exponential, dexponential, cexponential, lexponential, logistic, glogistic, gompertz, ldl, qdl, qdq, fourier, dfourier, gaussian, dgaussian, emax, gemax); default expo
SENSE = *string token*	Sense of curve (right, left); default righ
ORIGIN = *scalar*	Constrained origin; default *
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate); default comm
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default as in previous TERMS statement, or 3 if no TERMS given
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical); default *
FPROBABILITY = *string token*	Printing of probabilities for variance ratios (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob

Parameter

formula	Explanatory variate, list of variate and factor, or variate*factor

Description

FITCURVE provides a convenient way of fitting various standard curves. The response variate must be specified beforehand, using the MODEL directive in the usual way. The parameter of FITCURVE can be set just to a variate that supplies the x-values for the curve, if you simply want to fit a single curve. You can also include a factor if you want to fit separate curves for different groups of the observations: these are then *parallel curves*. The interaction between the variate and the factor can also be included, representing curves constrained to have common nonlinear parameters but separate linear parameters for each level of the factor. Finally, if the NONLINEAR option is set to separate as well as including the interaction, separate curves are fitted for each level, with only the estimate of variability being pooled.

The CURVE option specifies which of the standard curves is to be fitted. For some of these, the SENSE option lets you choose between alternative forms. Before describing the curves in detail, here is a list for convenient reference:

Exponential

exponential	$y = \alpha + \beta \times \rho^x + \varepsilon$
dexponential	$y = \alpha + \beta \times \rho^x + \gamma \times \sigma^x + \varepsilon$
cexponential	$y = \alpha + (\beta + \gamma \times x) \times \rho^x + \varepsilon$
lexponential	$y = \alpha + \beta \times \rho^x + \gamma \times x + \varepsilon$

Logistic

logistic	$y = \alpha + \gamma / (1 + \exp(-\beta \times (x - \mu))) + \varepsilon$
glogistic	$y = \alpha + \gamma / (1 + \tau \times \exp(-\beta \times (x - \mu)))^{1/\tau} + \varepsilon$
gompertz	$y = \alpha + \gamma \times \exp(-\exp(-\beta \times (x - \mu))) + \varepsilon$
emax	$y = \alpha + \gamma / (1 + \exp(-\beta \times (\log(x) - \mu))) + \varepsilon$
gemax	$y = \alpha + \gamma / (1 + \tau \times \exp(-\beta \times (\log(x) - \mu)))^{1/\tau} + \varepsilon$

Rational functions

ldl	$y = \alpha + \beta / (1 + \delta \times x) + \varepsilon$
qdl	$y = \alpha + \beta / (1 + \delta \times x) + \lambda \times x + \varepsilon$
qdq	$y = \alpha + (\beta + \gamma \times x)/(1 + \delta \times x + \eta \times x^2) + \varepsilon$

Fourier

fourier	$y = \alpha + \beta \times \sin(2\pi \times (x - \eta) / \omega) + \varepsilon$
dfourier	$y = \alpha + \beta \times \sin(2\pi \times (x - \eta) / \omega) + \gamma \times \sin(4\pi \times (x - \eta) / \omega) + \varepsilon$

Gaussian

gaussian	$y = \alpha + (\beta / \sqrt{(2\pi\sigma^2)}) \times \exp(-(x - \mu)^2/(2\sigma^2)) + \varepsilon$
dgaussian	$y = \alpha + (\beta / \sqrt{(2\pi\sigma^2)}) \times \exp(-(x - \mu)^2/(2\sigma^2)) + (\gamma / \sqrt{(2\pi\sigma^2)}) \times \exp(-(x - \nu)^2/(2\sigma^2)) + \varepsilon$

The four exponential curves each arise as solutions of linear ordinary differential equations. These represent processes that increase exponentially with time, for example, or that increase with a law of diminishing returns (that is, for which the rate of increase decreases with time).

The default setting of the CURVE option is `exponential`, corresponding to the "asymptotic regression" or Mitscherlich curve. The model has only one nonlinear parameter, ρ, which defines the rate of exponential increase or decrease. FITCURVE estimates the other parameters by linear regression at each stage of an iterative search for the best estimate of ρ. The values of the explanatory variate are automatically scaled to avoid any computational problems near the boundary of the allowed values of ρ. By default, ρ is restricted to the range $0<\rho<1$, giving a curve corresponding to the law of diminishing returns. The alternative is $\rho>1$, which can be requested by setting the SENSE option to `left`: for all the exponential curves, SENSE=left corresponds to a curve whose asymptote is to the left – that is, as X decreases to $-\infty$. If Genstat finds that a better fit is obtained by the opposite sense to the one specified, the sense is reversed and a warning is printed. The parameter α is the asymptote – to the right if $\rho<1$ and to the left if $\rho<1$; β is the range of the curve between the value at $X=0$ and the asymptote.

The double exponential curve also has two forms: you can choose either $0<\rho<1$ and $0<\sigma<1$ or $\rho>1$ and $\sigma>1$, by using the SENSE option as for the exponential curve. The fitting process is unlikely to find a satisfactory solution for this curve unless there are enough data to estimate both components separately: there should be at least four points for which the fast component is larger than the slow component; the fast component corresponds to the smaller of ρ and σ when `SENSE=right`, or to the larger of ρ and σ when `SENSE=left`.

Two limiting cases of the double exponential are provided as special curves. The critical exponential curve can take a variety of shapes like the double exponential, whereas the line-plus-exponential curve is an exponential curve with a non-horizontal asymptote. Again here, the

constraint on the parameter ρ depends on the setting of the SENSE option as for the exponential curve.

Another type of standard curve is sigmoid and monotonic, and is often used to model the growth of biological subjects. There are five types of these growth curves in Genstat, each a logistic of some sort. The first type is the generalized logistic without any constraints. In the equation above, α is one asymptote, to the right or to the left according to whether β is positive or negative; μ is the point of inflexion for the explanatory variable; β is a slope parameter; τ is a power-law parameter; and α+γ is the other asymptote. To fit this curve you need data for the steep central part and for both flat parts.

There are two special cases of the generalized logistic. The ordinary logistic curve is sometimes known as the autocatalytic or inverse exponential curve. The same curve can be rewritten in several different forms, so you should be alert for concealed equivalences of apparently different curves: otherwise you might be tempted to use FITNONLINEAR, which would be less efficient. The other special case is the Gompertz curve. It is non-symmetrical about the inflexion, X=μ, and has asymptotes at Y=α and Y=α+γ.

You can fit these three growth curves to data in which Y decreases as X increases. For the logistic and generalized logistic curves, you are not allowed to constrain the sense of the curve by the SENSE option. This is because the sense depends on both the parameters β and γ. In fact, the logistic curve with parameters α, β, γ and μ is the same as the logistic curve with parameters (α+γ), −β, −γ and μ; Genstat will report only one of the two possible versions. For the Gompertz curve, you can set SENSE=left to specify the upside-down Gompertz curve corresponding to γ<0; otherwise γ is constrained to be positive. When the sign of γ is changed for a response Y that increases with X, the sign of β will also change so that the curve remains an ascending one, and similarly for descending curves. The interpretation of SENSE=left thus depends on the shape of the data; for ascending curves it means that the asymptote is reached more slowly to the left than to the right, but for descending curves it means the opposite.

The final two sigmoid curves, Emax and generalized Emax, are similar to the logistic and generalized logistic except that their equations involve log(x) instead of x. They are usually used to model decreasing relationships with the parameter β in the equation negative, but Genstat will allow increasing relationships with these curves too.

The three rational functions are ratios of polynomials. The linear-divided-by-linear curve is a rectangular hyperbola, which occurs for example as the Michaelis-Menten law of chemical kinetics. The quadratic-divided-by-linear curve is a hyperbola with a non-horizontal asymptote. The quadratic-divided-by-quadratic curve is a cubic curve having an asymmetric maximum falling to an asymptote. The SENSE option is ignored for all three rational functions.

Fourier curves are trigonometric functions, involving the sine function in Genstat's implementation, used to model periodic behaviour. Sometimes the wavelength or period ω is a known constant, such as 2π radians (or 360 degrees), 24 hours or 12 months; the models are then linear and should be fitted by linear regression using the FIT directive, instead of by FITCURVE. The parameters β and γ are the amplitudes of the components of the curve. The SENSE option is ignored for Fourier curves.

The Gaussian curve is a bell-shaped curve like the Normal probability density. The double Gaussian is a sum of two overlapping curves of this type, and arises for example in spectography. The parameter α is usually called the *background*, and the parameters μ and ν are the peaks. The parameter σ is the standard deviation: for the double Gaussian, FITCURVE can deal only with the case of equal standard deviation for the two components. The parameters β and γ represent the strength of a spectrographic signal in each component, excluding the background. The SENSE option is ignored for Gaussian curves.

The PRINT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE and FPROBABILITY options are as for the FIT directive, and SELECTION differs only because FITCURVE caters only for the Normal distribution.

You can constrain the exponential and rational curves to pass through a given point. The ORIGIN option specifies a value for the response variate corresponding to a zero value of the explanatory variate; to specify the response for another value of the explanatory variate you would need to modify the explanatory variate beforehand.

Another way of constraining the curves is by setting the CONSTANT option to omit the constant term. This parameter represents an asymptote of each curve. To constrain the asymptote to be other than 0, you should put the value that you require into every element of the variate in the OFFSET option of the MODEL directive. The constant cannot be omitted from the Gompertz fitted with SENSE=left, nor (for any curve) if the origin is constrained, nor if parallel curves are fitted.

You can use the WEIGHTS option of the MODEL directive to supply a variate of weights for the units. You can also supply a symmetric matrix of weights, for example to allow for covariances between units. However, if the model contains an explanatory factor, pairs of units with different factor levels must have zero covariances.

You can modify a model fitted by FITCURVE by using the ADD, DROP or SWITCH directives as for models fitted by the FIT directive, provided the alterations produce a model that would be allowed in FITCURVE: that is, it must contain one variate, or one variate and one factor, or one variate and one factor and their interaction. The NONLINEAR options of the ADD, DROP and SWITCH directives have the same effect as the NONLINEAR option of FITCURVE. Thus you can compare curves between groups of a factor, assessing for example whether they are parallel. The accumulated setting of the PRINT option of these directives allows you to summarize the results. The RDISPLAY directive can be used to display further output following FITCURVE, and results can be copied into Genstat data structures using the RKEEP directive.

If you have group and fit models with common values of the nonlinear parameters across the groups, FITCURVE is unable provide standard errors for the linear parameters. If you need these, you can use the procedure RCURVECOMMONNONLINEAR.

Options: PRINT, CURVE, SENSE, ORIGIN, NONLINEAR, CONSTANT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, SELECTION.
Parameter: unnamed.

Action with RESTRICT

You can restrict the units that Genstat will use for fitting the curve by putting a restriction on the response or offset variates (defined by the MODEL directive), or on the explanatory variate or factor in the FITCURVE statement. However, you are not allowed to have different restrictions on the different vectors.

See also

Directives: MODEL, TERMS, RDISPLAY, RKEEP, RKESTIMATES, RCYCLE, RFUNCTION, ADD, DROP, SWITCH, FIT, FITNONLINEAR.
Procedures: RCHECK, RCURVECOMMONNONLINEAR, RGRAPH, RDESTIMATES, MICHAELISMENTEN, NLAR1, HGNONLINEAR, RQNONLINEAR, RQUADRATIC, RSCHNUTE, DFUNCTION.
Genstat Reference Manual 1 Summary section on: Regression analysis.

FITNONLINEAR

Fits a nonlinear regression model or optimizes a scalar function.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, grid); default mode, summ, esti or grid if NGRIDLINES is set
CALCULATION = *expression structures*	Calculation of fitted values or of explanatory variates involving nonlinear parameters; default * (valid only if OWN set)
OWN = *scalar*	Option setting for OWN directive if this is to be used rather than CALCULATE; default * requests CALCULATE to be used
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit for expansion of model terms; default as in previous TERMS statement, or 3 if no TERMS given
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
NGRIDLINES = *scalar*	Number of values of each parameter for a grid of function evaluations; default *
SELINEAR = *string token*	Whether to calculate s.e.s for linear parameters (yes, no); default no
INOWN = *identifiers*	Setting to be used for the IN parameter of OWN if used in place of CALCULATE; default *
OUTOWN = *identifiers*	Setting to be used for the OUT parameter of OWN if used in place of CALCULATE; default *

Parameter

formula	List of explanatory variates and/or one factor to be used in linear regression, within nonlinear optimization

Description

FITNONLINEAR can fit nonlinear regression models, or optimize a general function. However, you should check first that the model is not included in the linear and generalized linear models that can be fitted by FIT or the standard curves provided by FITCURVE, for these are much more efficient. Because the methods used by FITNONLINEAR are very general, they are neither as robust nor as automatic as, for example, the method that is used for fitting linear models.

It is better also to use FIT for nonlinear models with separable linear parameters (generalized nonlinear models), because there are fewer constraints on the model specification. FIT differs from FITNONLINEAR as follows. FIT takes account of the settings of the LINK and EXPONENT options of MODEL (but FITNONLINEAR does not). Furthermore, in FIT the parameter may be set to a general model formula, including more than one factor, interactions and functions. However, FIT cannot be used for optimisation of general functions, nor for models with no linear parameters.

Nonlinear methods make use of iterative optimization algorithms, designed to search for the minimum value of a function as the parameters vary; for nonlinear regression models, the function involved is the deviance, or minus twice the log-likelihood ratio, so the algorithm searches for the maximum-likelihood solution. It is often necessary to provide the algorithm with good starting values, to set bounds on the parameter values, and sometimes even to define the initial direction of search. Optimization is easiest with few parameters, approximately quadratic functions, small correlations between parameters, and good initial parameter estimates. Where possible, you can effectively reduce the number of parameters to be optimized by separating linear and nonlinear ones: that is, you can first fit the linear parameters, and treat the resulting residual sums of squares as functions of the nonlinear parameters alone. Problems with optimization methods are most likely to arise if you neglect the parameterization of the function. You can often transform the parameters to make the function nearly quadratic; after finding a solution, you can then use the RFUNCTION directive to estimate the original parameters. Another source of difficulty is if you try to fit inappropriately many parameters. For advice on reformulating functions to speed up optimization, see Ross (1990).

Before using FITNONLINEAR you must use the MODEL directive with either the Y parameter set to response variate, or the FUNCTION option set to a scalar that is to store the value of a general function to be optimized. You must also use the RCYCLE directive to specify the nonlinear parameters. The TERMS directive can be used as in linear regression, to list the explanatory variables to be used in modelling. The model calculations themselves are provided in expression structures which are supplied by the CALCULATION option of FITNONLINEAR. You can modify the model using the ADD, DROP and SWITCH directives as usual, except that you must give a TERMS statement first. You can use the RDISPLAY and RKEEP directives to display or save the results. The RCHECK procedure can be used to assess the fit of the model, and RGRAPH can display the fit with respect to some specified variate.

Genstat fits nonlinear regression models by maximum likelihood. The likelihood is usually from a distribution in the exponential family; this is specified using the DISTRIBUTION option of the MODEL directive. The settings of the LINK and EXPONENT options of the MODEL directive are ignored, and you are not allowed to set the GROUPS option; other options and parameters are as in linear regression.

The RCYCLE directive allows you to select the algorithm (of the three available) to be used to fit the model; these work with numerical differences and so do not require you to specify derivatives. The default algorithm is a modified Gauss-Newton method. This takes advantage of the fact that the likelihood function can be expressed as a sum of squares. However, you cannot use it for minimizing a general function. The second algorithm, a modified Newton method, is requested by setting option METHOD=Newton in the RCYCLE statement. This can be used for any nonlinear model. The third algorithm is a modified Fletcher-Powell method, specified by setting METHOD=Fletcher. In fact, this is similar to the Newton method, with an

occasional step in the search being determined by the Fletcher-Powell algorithm rather than by the Newton algorithm. The modification in all these methods is to use estimated numerical differences instead of evaluating derivatives. In nonlinear regression problems, particularly ones with separable linear parameters, specification of the derivatives would be very complex, and so it is much more convenient to estimate them numerically.

You must set the PARAMETER parameter of RCYCLE to the identifiers of scalars that will be used to represent the nonlinear parameters in the model calculations. There must be at least one nonlinear parameter. There is no formal upper limit on the number of nonlinear parameters, but the greater the number of parameters the longer the time required for the search and the smaller the chance of finding a satisfactory solution.

You can set the LOWER and UPPER parameters of RCYCLE to provide fixed bounds for each parameter. By default, the values $\pm 10^9$ are used. Where possible you should always set bounds, particularly to avoid such problems as attempting to take the log of a negative number. You can incorporate more general constraints as logical functions within the calculations. For example you could compute an extra term

```
(Constr > 0) * K * Constr
```

to impose a penalty on exceeding the constraint, controlled by setting different values of K. Often, the best way to impose a constraint is to reparameterize. For example, if a parameter α must be positive, you could replace α by $\exp(\beta)$, and allow β to take any value.

The STEPLENGTH parameter of RCYCLE can be used to provide initial step lengths for the search. By default the step length is 0.05 times the initial value of the corresponding parameter, or precisely 1.0 if the initial value is zero. If you set a step length to zero, Genstat treats the corresponding parameter as being fixed at its initial value. This allows complex problems in many dimensions to be tackled in stages, optimizing some parameters with others fixed, and then optimizing the others in turn.

By default, the initial value of a parameter is taken to be the current value of the scalar that represents it in the calculation, or 1.0 if the value is missing. Other values can be specified using the INITIAL parameter of RCYCLE.

If you can calculate a range within which you expect a parameter to lie, you should choose a step length of about 1% of the width of the range. If the steps are too small, numerical differencing may not work; if they are too large, gradients may be unreliable and you may get premature convergence. Genstat tests convergence by the relationship of final adjustments to step lengths.

The more parameters there are to estimate, and the more scattered are the data, the more iterations are required to find the optimum. The maximum number of iterations is set to 30 by default, but you can reset this with the MAXCYCLE option of RCYCLE. However, if convergence fails with a given setting of MAXCYCLE, you should check the data and consider reparameterizing the model before you indiscriminately increase the number of iterations.

Genstat prints a warning when convergence fails. The only sections of output that are then available are the residual degrees of freedom, the residual deviance, the fitted values, and the parameter estimates (without standard errors) for the current cycle. The EXIT parameter of the RKEEP directive allows you to obtain a numerical code indicating why convergence failed.

Many of the options of FITNONLINEAR are the same as those of FIT: PRINT, CONSTANT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY and SELECTION are all as in FIT. The grid setting of PRINT is used with the NGRIDLINES option. If you set NGRIDLINES to n, say (with $n \geq 2$), FITNONLINEAR evaluates the likelihood at a grid of values of the nonlinear parameters, and does not search for an optimum. For each parameter, the distance between the upper and lower bounds (set by the RCYCLE directive) will be divided into $(n-1)$ equal parts, defining a rectangular grid with n gridlines in each dimension. By setting some upper and lower bounds equal, you can look at the behaviour of the function with respect to a few parameters at a time. The default setting of the PRINT option is grid in this case, and produces a display of

the function values. Other settings of the PRINT option are then ignored. The calculated grid of values is available from the GRID parameter of the RKEEP directive, and can be used to produce pictures of the surface for example with the DCONTOUR or DSURFACE directives.

You must set one of the CALCULATION and OWN options to define how the nonlinear parameters are included in the model. The CALCULATION option does this by a list of one or more expressions. The expressions are evaluated in turn at every step of the estimation process, just as if they had been given in a sequence of CALCULATE statements. For example:

```
EXPRESSION Diffuse[1]; \
  VALUE=!E(Xl,Xr=NORMAL((H+1,-1*X)/SQRT(2*D*T))
& Diffuse[2]; VALUES=!E(Z=Xl+Xr-1)
FITNONLINEAR [CALCULATION=Diffuse[1,2]] Z
```

Here, the CALCULATION option is set to the two expressions Diffuse[1] and Diffuse[2], to define a model for one-dimensional diffusion. Alternatively, you can set the OWN option to specify that the calculation is to be done by executing your own source code, called by a version of the subroutine G5XZXO, as for the OWN directive. Generally, using OWN is likely to be worthwhile only when calculations are very extensive, or when a particular function is needed often. The setting of the OWN option will be passed to G5XZXO in the same way as the setting of the SELECT option of the OWN directive is passed to G5XZXO. The INOWN and OUTOWN options then define data structures to provide the input and store the output from FITNONLINEAR.

There are three ways of using FITNONLINEAR. The first provides the most efficient method when the model is linear in some of the parameters. However, this can be used only if the data are Normally distributed, or if they follow a Poisson distribution and the model contains only one explanatory variable and no constant term. The linear parameters are fitted by a linear regression of the response variate (specified by the parameter of the MODEL statement) on the variates listed by the parameter of FITNONLINEAR. At least one of these variates must depend on the nonlinear parameters in the model but they need not all do so. You can define how to calculate the variates from the nonlinear parameters either by the CALCULATION option or by the OWN, INOWN and OUTOWN options of FITNONLINEAR. The parameter of FITNONLINEAR may include variates that are not changed by the calculations as well as those that are. One factor may also be included so that a separate constant is fitted for each level, giving a set of parallel nonlinear regressions. You cannot include interactions between a variate and a factor, as is allowed with FIT or FITCURVE; nor can you include POL, REG, COMPARISON, SSPLINE or LOESS functions, nor interactions between variates as allowed with FIT. However, procedure FITPARALLEL allows you to assess the various ways in which nonlinear models can be non-parallel. If there is a constant in the linear regression, as specified by the CONSTANT option, the factor will be parameterized in terms of differences from the first level – as in linear regression. If you set CONSTANT=omit, the actual constants are fitted; there is no need to set option FULL of the TERMS directive which is ignored in nonlinear models. If you have specified an offset variate using the MODEL directive, its values can also be modified by the calculations, and depend on the parameters. By default, standard errors are calculated only for nonlinear parameters. To obtain standard errors for the linear parameters as well, you can set option SELINEAR=yes. Then, after the optimum has been found, Genstat increases the number of dimensions to include the linear parameters and estimates the rate of change of the likelihood in all the dimensions.

If there are no linear parameters in the model, or if the distribution is not Normal or Poisson, you should no longer use the parameter of FITNONLINEAR. Instead you should set the FITTEDVALUES parameter in the MODEL statement to the identifier of a variate that is to contain the fitted values for any set of values of the nonlinear parameters. Then define how to calculate the fitted values from the nonlinear parameters and the explanatory variates, using either the CALCULATION or the OWN option of FITNONLINEAR. The distribution can now be any of those available from the DISTRIBUTION option of the MODEL directive, taking account of the settings

of the DISPERSION and WEIGHTS options of the MODEL directive. The multinomial distribution is used rather differently from the others: it is for fitting distributions. The DISTRIBUTION directive provides a wide range of standard distributions, and is more convenient and efficient than FITNONLINEAR for these; but FITNONLINEAR allows you to fit other distributions. To specify and fit your own distribution, you should supply as response variate a set of counts of observations falling into a series of groups; the fitted values should then be a set of expected counts for the groups, calculated from the distribution being considered. The resulting multinomial likelihood is the same as that of the Poisson distribution, but with the constraint $\Sigma f_i = M$, where M is the sum of the counts.

The third method allows you to minimize a general function. (You can still use this to fit statistical models by supplying the deviance, which is minus twice the log-likelihood ratio.) To minimize a function, you need to start with a MODEL statement that has no response variate, but where the FUNCTION option is set to a scalar. You then specify the parameters with the RCYCLE directive as before, and perform the minimization with FITNONLINEAR, supplying an expression that calculates the function from the parameters and places the result into the scalar. When you are minimizing a general function in this way, some of the output from FITNONLINEAR is different. Genstat ignores the accumulated and fittedvalues settings of the PRINT option, and the deviance and summary settings display only the minimum function value. The correlation setting displays the inverse of the estimated matrix of second derivatives of the function with respect to the parameters, scaled by the diagonal values. Similarly, in place of the standard errors usually displayed by the estimates setting, Genstat prints the square roots of the diagonal values of twice the inverse of the second-derivative matrix. These can give a useful indication of the form of the function near the minimum. If the function is a deviance you can interpret these as asymptotic standard errors and correlations (not scaled by an estimate of dispersion). For a general function, the "s.e." can be interpreted as the approximate change in a parameter required to increase the function by 1.0 starting from the minimum. Genstat ignores the CONSTANT option of the FITNONLINEAR directive for general functions, and you must not set the parameter. Similarly, the WEIGHTS and OFFSET options of the MODEL directive are ignored, and the GROUPS option must not be set. The only parameters of the RKEEP directive that are available are ESTIMATES, SE, INVERSE, EXIT, GRADIENTS and GRID. The minimum value of the function is of course available in the scalar specified by the FUNCTION option of the MODEL directive.

Options: PRINT, CALCULATION, OWN, CONSTANT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, SELECTION, NGRIDLINES, SELINEAR, INOWN, OUTOWN.
Parameter: unnamed.

Action with RESTRICT

You can restrict the units that Genstat will use for fitting the model by putting a restriction on any of the vectors involved in the MODEL statement (response variate, weight variate, offset variate or variate of binomial totals), or on any explanatory variate or factor. However, you are not allowed to have different restrictions on the different vectors.

Reference

Ross, G.J.S. (1990). *Nonlinear Estimation*. Springer-Verlag, New York.

See also

Directives: EXPRESSION, MODEL, RDISPLAY, RKEEP, RCYCLE, RFUNCTION, FIT, FITCURVE.

Procedures: NLAR1, HGNONLINEAR, RQUADRATIC, MINIMIZE, MIN1DIMENSION, SIMPLEX.

Genstat Reference Manual 1 Summary section on: Regression analysis.

FKEY

Forms design keys for multi-stratum experimental designs, allowing for confounded and aliased treatments.

Options

BASICFACTORS = *factors*	Factors indexing the units of the design
ADDEDFACTORS = *factors*	Factors to be allocated to the units of the design
KEY = *matrix*	Stores the design key (ADDEDFACTORS × BASICFACTORS)
INKEY = *matrix*	Can be used to input existing allocations for some of the added factors
HIERARCHIES = *matrix*	Can be used to specify that some of the factors must be constant within each combination of levels of other factors; the matrix has a row for each added factor and columns first for the basic factors and then for the added factors, ones in the entries where the row factor must be constant within the combinations of the column factors, zero elsewhere
SEED = *scalar*	Can provide a seed to generate a random permutation of the sets of basic effects that may be allocated to each added factor, thus producing design randomly selected from all those that might be possible; default * i.e. no permutation
ROWPRIMES = *variate*	Prime numbers for the rows of the KEY matrix
COLPRIMES = *variate*	Prime numbers for the columns of the KEY matrix
ROWMAPPINGS = *variate*	Mappings from the rows of the KEY to the TREATMENTFACTORS
COLMAPPINGS = *variate*	Mappings from the columns of the KEY to the BLOCKFACTORS
SAVE = *identifier*	Structure to save all the information about the formation of the design; this can then be input later to give a different design (if possible) with the same properties

Parameters

REQUIRED = *formula structures*	Formulae each defining a list of terms that are to be estimated in the analysis
NONNEGLIGIBLE = *formula structures*	Formulae each specifying terms that cannot be ignored in the context of the corresponding REQUIRED formula

Description

Design keys can be used in the GENERATE directive to generate values of treatment factors from block factors. They also provide the basis of the representation used to store the repertoire of designs obtainable from procedure AGDESIGN (see Payne and Franklin 1994). This covers a range of standard situations, but cannot allow for every eventuality. FKEY allows you to form keys for other circumstances and, if these are likely to occur frequently, you can extend or replace the standard repertoire using procedure FDESIGNFILE.

The assumption in FKEY is that the units of the design are indexed by a set of factors known as the *basic* factors. The key allows the values of another set of factors, known here as the *added* factors, to be calculated from the basic factors. These factors are listed using the BASICFACTORS and ADDEDFACTORS options. They must all have been declared previously as factors, and their

numbers of levels must have been defined. Usually the basic factors are the factors that will be used to define the block formula of the design (for example, blocks, plots, rows, columns, subplots and so on) and the added factors are the treatment factors, but in partial replicates, for example, the basic factors may be the treatment factors and the added factors the block factors.

If the basic and added factors all have prime numbers of levels the key is saved, by the `KEY` option, as a matrix with a row for each added factor and a column for each basic factor. However, if the levels are not all prime, `FKEY` will break up factors that do not have prime numbers of levels into "pseudo-factors". Thus, a factor with six levels will be represented by the combinations of levels of two pseudo-factors, one with two levels and one with three levels. When pseudo-factors are required for the added factors, the `ROWPRIMES` option can be used to save a variate storing the (prime) number of levels corresponding to each row of the key, and the `ROWMAPPINGS` option can save a variate with an element for each row containing the number of the corresponding added factor. So, if we had two added factors, one with five and one with six levels, the `ROWPRIMES` variate might contain the values 5, 2 and 3, and the `ROWMAPPINGS` variate the values 1, 2 and 2. The second added factor (with six levels) would then be represented by two pseudo-factors, corresponding to the second and third rows of the key. The `COLPRIMES` and `COLMAPPINGS` options can similarly save details of the pseudo-factors required for basic factors with non-prime numbers of levels. The variates saved by `ROWPRIMES`, `COLPRIMES`, `ROWMAPPINGS` and `COLMAPINGS` can be used in the `AKEY` procedure, together with the key, to form the added factors automatically without the need to worry about the pseudo-factoring.

The main properties of the design are derived from the `REQUIRED` and `NONNEGLIGIBLE` parameters. Suppose we have a block design containing three blocks of nine plots. The experiment is to have three treatment factors, `A`, `B` and `C`, and these will be the added factors. The design has a block structure of plots nested within blocks

```
Blocks/Plots
```

In the analysis we wish to be able to estimate all main effects and interactions of the factors `A`, `B` and `C`, except the three-factor interaction `A.B.C`; these terms are specified by the formula structure supplied using the `REQUIRED` parameter. The `NONNEGLIGIBLE` parameter specifies model terms that cannot be ignored in the analysis: that is, the model terms with which these required terms cannot be confounded. Here we have the main effect `Blocks` and all main effects and interactions of the factors `A`, `B` and `C`. To form the design key `K`, we thus need to put

```
FACTOR [NVALUES=27; LEVELS=3] Block,A,B,C
& [LEVELS=9] Plot
FKEY [BASIC=Block,Plot; ADDED=A,B,C; KEY=K; \
  COLPRIMES=Bplev; COLMAPPINGS=Bmap] \
  REQUIRED=!f(A*B*C-A.B.C); NONNEGLIGIBLE=!f(Block+A*B*C)
```

If the design has more than two strata suitable for the estimation of treatment effects, the `REQUIRED` and `NONNEGLIGIBLE` parameters can specify lists of formulae, in parallel, one pair of formulae for each stratum. Each `REQUIRED` formula specifies the terms that must be estimated in one of the strata (or in a stratum below it), and the corresponding `NONNEGLIGIBLE` formula specifies the terms that cannot be ignored there. Suppose we put

```
FACTOR [NVALUES=81; LEVELS=3] Block,Wplot,A,B,C,D,E
& [LEVELS=9] Subplot
FKEY [BASIC=Block,Wplot,Subplot; ADDED=A,B,C,D,E; KEY=K; \
  COLPRIMES=Bplev; COLMAPPINGS=Bmap] \
  REQUIRED=!f((A+B+C)*(A+B+C)),!f((A+B+C+D+E)*(A+B+C+D+E));\
  NONNEGLIGIBLE=!f(Block+Block.Wplot),!f(Block)
```

Here we have a block formula

```
Block / Wplot / Subplot1
```

which produces three strata

```
Block + Block.Wplot + Block.Wplot.Subplot
```

The first formula in the REQUITRED list `!f((A+B+C)*(A+B+C))`, in parallel with the formula `!f(Block+Block.Wplot)` in the NONNEGLIGIBLE list, indicates that we do not want the main effects or two-factor interaction of factors A, B and C to be confounded with each other nor with Block or Block.Wplot; this ensures that they will be estimated in the Block.Wplot.Subplot stratum. The second pair of formulae, `!f((A+B+C+D+E) * (A+B+C+D+E))` and `!f(Block)`, indicate that we want to estimate the main effects and two-factor interactions of all the five treatment factors A, B, C, D and E in the Block.Wplot stratum or below; in effect this means that we are willing to have D and E and any of their interactions estimated in the Block.Wplot stratum.

The algorithm that FKEY uses to construct the key is based on the method developed by Franklin & Bailey (1977), Franklin (1985) and Kobilinsky (1995). Essentially this considers the possible orthogonal sets of contrasts amongst the main effects and interactions of the basic factors, and tries in turn to find a feasible set against which to confound each added factor. Often there are several feasible ways in which this can be done. To avoid FKEY selecting the same key every time, you can set the SEED option to an integer that will be used to generate a random permutation of the order in which the sets of basic contrasts are considered, thus producing design randomly selected from all those that might be possible; by default no permutation takes place. Alternatively, you can use the SAVE option to save all the information about the formation of the design; this can then be input later to provide the next possible key (if available) with the requested properties.

In a multi-stratum design, you may wish to insist that some factors are applied to complete units of one of the strata. This can be done using the HIERARCHIES option, which allows you to indicate that some of the added factors must be constant within each combination of levels of other factors. These constraints are specified, if required, by supplying a matrix with a row for each added factor and columns first for the basic factors and then for the added factors. The matrix contains ones in the entries where the row factor must be constant within the combinations of the column factors, and zeros elsewhere.

FKEY can also be used to extend an existing design, by allocating further factors to the units. The existing key should then be input using the INKEY option, with zeros in the rows for the new added factors.

FKEY can form keys for small designs fairly quickly, but for complicated arrangements you may find that it takes some time to check the various possibilities.

Options: BASICFACTORS, ADDEDFACTORS, KEY, INKEY, HIERARCHIES, SEED, ROWPRIMES, COLPRIMES, ROWMAPPINGS, COLMAPPINGS, SAVE.
Parameters: REQUIRED, NONNEGLIGIBLE.

References

Franklin, M.F. (1985). Selecting defining contrasts and and confounded effects in p^{n-m} factorial experiments. *Technometrics*, **27**, 165-172.

Franklin, M.F. & Bailey, R.A. (1977). Selection of defining contrasts and confounded effects in two-level experiments. *Applied Statistics*, **26**, 321-326.

Kobilinsky, A. (1995). *PLANOR: Programme de Génération Automatique de Plans d'Expériences Réguliers*. INRA, Versailles.

Payne, R.W. & Franklin, M.F. (1994). Data structures and algorithms for an open system to design and analyse generally balanced designs. In: *COMPSTAT 94 Proceedings in Computational Statistics* (ed. R. Dutter & W. Grossmann), pp. 429-434. Physica-Verlag, Hiedelberg.

See also
Directives: AFMINABERRATION, GENERATE, FPSEUDOFACTORS.
Procedures: AKEY, ARANDOMIZE, ASAMPLESIZE, FACPRODUCT, FBASICCONTRASTS.
Genstat Reference Manual 1 Summary sections on: Design of experiments, Analysis of variance.

FLRV

Forms the values of LRV structures.

Options

`PRINT` = *string tokens*	Printed output required (`roots, vectors`); default `*` i.e. no printing
`NROOTS` = *scalar*	Number of roots or vectors to print; default `*` i.e. print them all
`SMALLEST` = *string token*	Whether to print the smallest roots instead of the largest (`yes, no`); default `no`
`TOLERANCE` = *scalar*	Tolerance for detecting zero roots

Parameters

`INMATRIX` = *matrices* or *symmetric matrices*	Matrices whose latent roots and vectors are to be calculated
`LRV` = *LRVs*	LRV to store the latent roots and vectors from each `INMATRIX`
`WMATRIX` = *symmetric matrices*	(Generalized) within-group sums of squares and products matrix used in forming the two-matrix decomposition; if any of these is omitted, it is taken to be the identity matrix, giving the usual spectral decomposition
`ILRV` = *LRVs*	LRV to store the imaginary parts of the latent roots and vectors arising from the decomposition of a non-symmetric matrix

Description

The `INMATRIX` parameter lists the matrices for which latent roots and vectors are to be calculated. If the `WMATRIX` parameter is not set, `FLRV` provides the solution of the *one-matrix eigenvalue problem*

$$AX = XL$$

A is usually an n-by-n symmetric matrix. XLX' is then the spectral decomposition of the symmetric matrix A. Here L is a diagonal matrix containing the n latent roots, or eigenvalues, of A ordered such that

$$l_1 \geq l_2 \geq \ldots \geq l_n$$

The columns of the n-by-n matrix X are the corresponding latent vectors, or eigenvectors. The matrix X is orthogonal:

$$X'X = XX' = I_n$$

The three options of `FLRV` control the printing of the results. You use the `PRINT` option to specify whether you want the roots or vectors to be printed. If you request the roots to be printed, the trace will be printed as well. By default nothing is printed. The `NROOTS` option governs how many of the roots and vectors are printed, while the `SMALLEST` option determines whether the largest or smallest roots, and corresponding vectors, are printed.

You can use the `LRV` parameter to save the latent roots and vectors, and the trace. You must declare these structures in advance if you want to save less than the full number of roots; otherwise, they are defined automatically, as `LRVs` with n rows. You can save a subset of the latent roots and vectors by supplying an LRV structure with fewer columns than rows. However this saves only the largest roots and the corresponding vectors. You cannot save the smallest roots directly, as the `SMALLEST` option applies only to printing. If you want to save the smallest roots, then you must save the complete set of roots and vectors, and extract the last columns of

the matrix, for example using qualified identifiers. These rules are the same as those applied in the directives for multivariate analysis.

Alternatively, A can be a square, unsymmetric, matrix. This again provides the solution of the eigenvalue problem

$$AX = XL$$

but now A is a square matrix of order n, L is a diagonal matrix of n latent roots, and X is a square matrix of order n containing the right latent vectors of A. The solution of this problem may produce some complex latent roots, occurring as complex conjugate pairs, in which case the corresponding latent vectors are also complex conjugate pairs. To accomodate this, `FLRV` has a parameter, `ILRV`, for specifying an LRV structure to store the imaginary parts of the latent roots and vectors (the real parts being stored by the `LRV` parameter). The `ILRV` parameter need not be set, but a warning message is then printed if any complex roots are produced. If all the latent roots are real, they are sorted into descending order, such that $l_1 \geq l_2 \geq ... \; l_n$, as in the symmetric case, but if some roots are complex they are ordered such that $|l_1| \geq |l_2| \geq ... \geq |l_n|$. To detect whether a latent root is real, Genstat checks whether imaginary part is close to zero; to allow for numerical imprecision the value is tested against $|l_1|$ multiplied by the valued supplied by the `TOLERANCE` option, by default 10^{-6}. The values saved by the `LRV` and `ILRV` parameters, however, are those generated by the algorithm, so procedures using `FLRV` may also need to test explicitly for zero roots. The `TOLERANCE` option and `ILRV` parameter are ignored if `INMATRIX` is set to a symmetric matrix. Percentage variations are printed only if all roots are real. The latent vectors x_i are normalized so that $x_i' x_i = 1$, but this is not sufficient to determine them uniquely since they can still be scaled by any (complex) scalar z such that $|z|=1$. The convention adopted in Genstat is to apply an additional scaling such that the largest element of each x_i is real and positive. The latent vectors are guaranteed to be orthogonal only when the matrix A is symmetric.

`FLRV` can also solve the *two-matrix eigenvalue problem*

$$AX = WXL$$

The symmetric matrix W is specified using the `WMATRIX` parameter. A is a symmetric matrix again specified by the `INMATRIX` parameter; if this is set to a square matrix, the `WMATRIX` parameter is ignored. L is a again diagonal matrix, and X a square matrix. Both A and W must have the same number of rows, n, and W must be positive semi-definite. Now the latent roots are the n elements of the diagonal matrix L and are the successive maxima of

$$l = (x'Ax) / (x'Wx)$$

where x is the corresponding column of the n-by-n matrix X, normalized so that $X'WX=I$. The two-matrix decomposition is particularly relevant for canonical variate analysis.

For either eigenvalue problem, the sum of the latent roots is stored in the element of the `LRV` labelled `'Trace'`. In the one-matrix problem, this is also the trace of the original matrix A; but for the two-matrix problem, it is the trace of $W^{-1}A$. Latent roots are often expressed as percentages of the trace.

The method used for the spectral decomposition of a symmetric matrix first reduces the matrix to tri-diagonal form using Householder transformations (Martin, Reinsch & Wilkinson 1968); this is followed by a QL algorithm for finding the eigenvalues and eigenvectors (Bowdler, Martin, Reinsch & Wilkinson 1968). The algorithm used for the unsymmetric eigenvalue problem is based on NAG Library subroutine F02EBF. The documentation of this routine should be consulted for a full discussion of the method and accuracy of the results (NAG 1994). If `INMATRIX` is set to a matrix `A` of order n which happens to be symmetric the results should be identical, up to the sign of the latent vectors, apart from small numerical discrepancies of the order of machine precision and dependent on n and the condition number of `A`. The two-matrix problem is solved using two spectral decompositions, each computed as for the first problem.

Options: `PRINT`, `NROOTS`, `SMALLEST`, `TOLERANCE`.
Parameters: `INMATRIX`, `LRV`, `WMATRIX`, `ILRV`.

References

Bowdler, H., Martin, R.S., Reinsch, C. & Wilkinson, J.H. (1968). The QR and QL algorithms for symmetric matrices. *Numerische Mathematik*, **11**, 293-306.

Martin, R.S., Reinsch, C. & Wilkinson, J.H. (1968). Householders tridiagonalisation of a symmetric matrix. *Numerische Mathematik*, **11**, 181-195.

Numerical Algorithms Group. (1994). F02EBF. *NAG Fortran Library Mark 15, Volume 5*. Oxford: Numerical Algorithms Group.

See also

Directives: LRV, MATRIX, SYMMETRICMATRIX, NAG, QRD, SVD.

Functions: EVALUES, EVECTORS.

Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Multivariate and cluster analysis.

FOR

Introduces a loop; subsequent statements define the contents of the loop, which is terminated by the directive ENDFOR.

Options

NTIMES = *scalar*	Number of times to execute the loop; default is to execute as many times as the length of the first parameter list or once if the first list is null
INDEX = *scalar*	Records the loop index
START = *scalar*	Defines an integer initial value for the loop index; default 1
END = *scalar*	Defines an integer final value for the loop index
STEP = *scalar*	Defines an integer amount by which to increase the index each time the loop is executed; default 1
VALUES = *variate*	Defines a set of values to be taken successively by the loop index (overrides START, END and STEP if these are specified too)

Parameters

	Any number of parameter settings of the form *identifier = list of data structures*; the *identifier* is set up as a dummy which is then used within the loop to refer, in turn, to the structures in the list

Description

The FOR loop is a series of statements that is repeated several times. The FOR directive introduces the loop and indicates how many times it is to be executed. In its simplest form FOR has no parameters, and the number of times is indicated by the NTIMES option. For example, the following loop calculates the mean of three sets of data stored in the file attached to channel 2:

```
FOR [NTIMES=3]
  READ [CHANNEL=2] X
  CALCULATE Mean = MEAN(X)
  PRINT Mean; DECIMALS=4
ENDFOR
```

The INDEX option allows you to record the loop index in a scalar. By default this is the number of the time that the loop is currently being executed. So, in the statement below, the index Count will take the values 1, 2 and 3.

```
FOR [NTIMES=3; INDEX=Count]
```

The options START, END and STEP allow you to define a loop index that does not start at one, and does not increase by one each time the loop is executed. They should all be set to integers; any non-integer value is rounded to the nearest integer. (Integer calculations are exact, so this avoids inaccuracies due to numerical round-off when loops are executed many times.) START specifies the INDEX value on the first time that the loop is executed (default 1). STEP defines how it changes between one time that the loop is executed and the next (default 1). So, for example, on the second time INDEX will be START + STEP. END provides an alternative way of specifying how many times to execute the loop – it stops when the next index will go beyond END. For example, the statement below

```
FOR [INDEX=Count; START=3; END=8; STEP=2]
```

defines a loop that will be executed three times, with the index variable Count taking the values 3, 5 and 7; the next value would be 9, which goes beyond 8. The default STEP is one. STEP can also be negative. So, this statement

```
FOR [INDEX=Count; START=3; END=-4; STEP=-2]
```

defines a loop that will be executed four times, with the index variable Count taking the values 3, 1, −1 and −3; the next value would be −5, which goes beyond −4. If you specify NTIMES as well as END, they must both define the same number of times to execute the loop.

The VALUES option allows you to specify an arbitrary sequence of values for the loop index, and these need not be integers. The setting is a variate. So, for example, here

```
VARIATE [VALUES=0, 0.5, 1, 1.5, 2, 1.5, 1, 0.5, 0] Cvals
FOR [INDEX=Count; VALUES=Cvals]
```

Count will first increase from 0 to 2 in steps of 0.5, and then decrease back down to 0. The number of values in the VALUES variate must be the same as the value supplied by NTIMES if both options are specified. VALUES overrides START, END and STEP if these are specified too.

The INDEX is defined automatically as a scalar if it has not already been declared. If VALUES is set, its default number of decimals is set to be the same as the number defined for the VALUES variate (see the DECIMALS parameter of the VARIATE and SCALAR directives), or to take the default number if no decimals have been defined for VALUES. Otherwise the default number of decimals is set to zero.

The parameters of FOR allow you to write a loop whose contents apply to different data structures each time it is executed. Unlike other directives, the parameter names of FOR are not fixed for you by Genstat: you can put any valid identifier before each equals sign. Each of these then refers to a Genstat dummy structure; so you must not have declared them already as any other type of structure. The first time that the loop is executed, they each point to the first data structure in their respective lists, next time it is the second structure, and so on. The list of the first parameter must be the longest; other lists are recycled as necessary.

If you specify parameters you do not need to specify NTIMES but, if you specify both, the value of NTIMES must be the same as the length of the first parameter list.

You can specify as many parameters as you need. For example

```
FOR Ind=Age,Name,Salary; Dir='descending','ascending'
  SORT [INDEX=Ind; DIRECTION=#Dir] Name,Age,Salary
  PRINT Name,Age,Salary
ENDFOR
```

is equivalent to the sequence of statements

```
SORT [INDEX=Age; DIRECTION='descending'] Name,Age,Salary
PRINT Name,Age,Salary
SORT [INDEX=Name; DIRECTION='ascending'] Name,Age,Salary
PRINT Name,Age,Salary
SORT [INDEX=Salary; DIRECTION='descending'] Name,Age,Salary
PRINT Name,Age,Salary
```

printing the units of the text Name, and variates Age and Salary, first in order of descending ages, then in alphabetic order of names, and finally in order of descending salaries.

You can put other control structures inside the loop. So, for example, you can have loops within loops.

When you are using loops interactively, you may find it helpful to use the PAUSE option of SET to request Genstat to pause after every so many lines of output. Another useful directive is BREAK, which specifies an explicit break in the execution of the loop.

Options: NTIMES, INDEX, START, END, STEP, VALUES.
Parameters: names defining the dummies used within the loop.

See also
Directives: `ENDFOR`, `EXIT`, `CASE`, `IF`.
Genstat Reference Manual 1 Summary section on: Program control.

FORECAST

Forecasts future values of a time series (synonym of TFORECAST).

Options

PRINT = *string tokens*	What to print (forecasts, limits, setransform, sfe); default fore,limi
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
ORIGIN = *scalar*	Number of known values to be incorporated; default 0
UPDATE = *string token*	Whether to update the forecast origin to the end of the new observations (yes, no); default no
NEWOBSERVATIONS = *variate*	Variate of length ≥ ORIGIN providing new values of the time series to be incorporated (must be set if ORIGIN > 0)
SFE = *variate*	Saves standardized forecast errors; default *
MAXLEAD = *scalar*	Maximum lead time i.e number of forecasts to be made; default * defines the number as the length of FORECAST variate
FORECAST = *variate*	Variate of length MAXLEAD to save forecasts of output series; default *
SETRANSFORM = *variate*	Saves standard errors of the forecasts (on transformed scale, if defined); default *
LOWER = *variate*	Saves lower confidence limits; default *
UPPER = *variate*	Saves upper confidence limits; default *
PROBABILITY = *scalar*	Probability level for confidence limits; default 0.9
COMPONENTS = *pointer*	Contains variates (of length ORIGIN + MAXLEAD) to save components of the forecast
SAVE = *identifier*	Save structure to supply fitted model; default * i.e. that from last model fitted

Parameters

FUTURE = *variates*	Variates (of length ORIGIN + MAXLEAD) containing future values of input series
METHOD = *string tokens*	How to treat future values of input series (observations, forecasts); default obse

Description

FORECAST was renamed as TFORECAST in Release 14 to emphasize its status as a time-series command. The earlier name (FORECAST) was retained to allow previous programs to continue to run, but this may be removed in a future release.

FORECAST can be used after ESTIMATE to forecast future values of a time series. In many applications of forecasting with univariate ARIMA models, you will need only a simple form of the directive. For example

```
FORECAST [MAXLEAD=10]
```

will cause Genstat to print forecasts for 10 lead times, that is, the next 10 time points after the end of your data. However, you must already have used ESTIMATE to specify the time series to be forecast, and the model to be used for forecasting. This information is supplied by the SAVE option; if SAVE is not specified, FORECAST uses the information from the most recent ESTIMATE statement. Once you have used ESTIMATE, you can give successive FORECAST statements to incorporate new observations of the time series, and to produce forecasts from the end of the new data.

If the time series is supplied with a known model (that is, one with all its orders and parameters specified) you can use ESTIMATE with option setting METHOD=initialize before you use FORECAST. This will carry out just sufficient calculations, in particular the regeneration of the model residuals, for FORECAST to be used. The model parameters will not be changed – not even the innovation variance. This setting of METHOD restricts the structures, such as parameter standard errors, that can be accessed using TDISPLAY and TKEEP after ESTIMATE. The SAVE structure created by using ESTIMATE with METHOD=initialize thus requires less space than that produced by the other settings.

The PRINT option controls the printed output, and the CHANNEL option allows this to be sent to another output channel.

The ORIGIN option specifies the number of new values to be incorporated before forecasting ahead from that point. Setting this to a positive value *n* indicates that *n* new observations are to be added onto the end of the series. These new observations must be supplied in a variate using the NEWOBSERVATIONS option, which must be of length $\geq n$. The standardized forecast errors for these new observations can be printed or saved in a variate of length *n* using the SFE option.

The UPDATE option specifies whether these new observations are to be incorporated internally onto the end of the time series and the internal pointer moved to the end of the new observations. If UPDATE=yes is used, then ORIGIN=0 in future calls to FORECAST will point to the end of the *n* new observations. If the default, UPDATE=no is used, then the internal pointer remains at the end of the original series.

The number of future values to be forecast is set by option MAXLEAD. These new values can be saved in a variate of length MAXLEAD using the FORECAST option.

The PROBABILITY option determines the width of the error limits on the forecast. It defines the probability that the actual value will be contained within the limits at any particular lead time. Note that the limits do not apply simultaneously over all lead times.

The SETRANSFORM option specifies a variate to store the standard errors that Genstat used in calculating the error limits of the forecasts, starting at lead time 1. These are the standard errors of the transformed series, according to the value of the Box-Cox transformation parameter; they are functions of the model only, not of the data.

The LOWER option specifies a variate to store the lower limits of the forecasts. This must be the same length as the FORECAST variate. The FORECAST directive puts the values of the lower limit into the variate, matching the forecasts in the FORECAST variate. The UPPER option similarly allows the upper limits to be saved. Note that the limits are constructed as symmetric percentiles, assuming Normality of the transformed time series. Similarly, the forecast is a median value – not necessarily the mode or the mean, unless the transformation parameter is 1.0.

The SFE option specifies a variate to save the standardized errors of the forecasts. These are the innovation values that are generated as each successive new observation is incorporated, divided by the square root of the TSM innovation variance. They provide a useful check on the continuing adequacy of the model. For example, excessively large values (compared to the standard Normal distribution) may indicate that you should revise the model. The variate must be the same length as the FORECAST variate. The FORECAST directive places values of the errors in the variate, matching the new observations in the FORECAST variate.

The parameters of FORECAST are relevant only when the time-series model incorporates explanatory variables, which requires a TRANSFERFUNCTION statement before the ESTIMATE statement. You use the FUTURE parameter to specify a list of variates, corresponding to the list of variates specified by the SERIES parameter of TRANSFERFUNCTION. These variates must all have the same length. They hold future values of the explanatory variables to be used either for constructing forecasts of the response variable, or for incorporating new observations in order to revise the forecasts. You can use the METHOD parameter when some or all of the future values of the explanatory variables are forecasts obtained using univariate ARIMA models. You can amend the error limits of the forecasts for the response variable to allow for the uncertainty in

these future values, but you need to assume that there is no cross-correlation between the errors in these predictions. The list of strings specified by the METHOD parameter indicates for each explanatory variable whether such an allowance should be made. The future values of a series are by default treated as known values if no corresponding ARIMA model is present, or if the transformation parameter of the ARIMA model is not equal to the value used in the regression model for that series. You can change the settings of the METHOD parameter in successive FORECAST statements.

The COMPONENTS option is also relevant only when the time-series model incorporates explanatory variables, and can be used to specify a pointer to variates in which you can save components of future values of the output series. There is a variate for each input component and for the output noise component. These variates correspond exactly to the variates that were specified by the FUTURE parameter for the input series, and by the FORECAST variate for the output series; corresponding lengths must match. The values that the variates hold can therefore be components of the forecasts of the output series, or can be new observations. The can be used to investigate the structure of forecasts.

If the input series ARIMA model and the transfer-function model have differing transformation parameters, then the METHOD option reverts to its default action of treating the values of any future input series as known quantities rather than forecasts.

Options: PRINT, CHANNEL, ORIGIN, UPDATE, NEWOBSERVATIONS, SFE, MAXLEAD, FORECAST, SETRANSFORM, LOWER, UPPER, PROBABILITY, COMPONENTS, SAVE.
Parameters: FUTURE, METHOD.

See also
Directives: TFORECAST.
Genstat Reference Manual 1 Summary section on: Time series.

FORMULA

Declares one or more formula data structures.

Options

VALUE = *formula*	Value for all the formulae; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the formulae in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the formulae
VALUE = *formula structures*	Value for each formula
EXTRA = *texts*	Extra text associated with each identifier

Description

The IDENTIFIER parameter lists the identifiers of the formulae that are to be declared. The formula data structure stores a Genstat formula. This can be used to define the model to be fitted in a statistical analysis. Its main use is to give a formula as the argument of a procedure.

Values can be assigned to the formulae by either the VALUE option or the VALUE parameter. The option defines a common value for all the structures in the declaration, while the parameter allows the structures each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

You can associate a text of extra annotation with each formula using the EXTRA parameter. If MODIFY is set to yes any existing attributes and values of the formulae are retained; otherwise these are lost.

For example:

```
FORMULA [VALUE=Drug*Logdose] Model
FORMULA BModel,Tmodel; \
  VALUE=!F(Litter/Rat),!F(Vitamin*Protein)
```

The construction !F(Litter/Rat) is an example of an unnamed formula.

The IPRINT option can be set to specify how the formulae will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

Options: VALUE, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUE, EXTRA.

See also

Directives: FCLASSIFICATION, REFORMULATE, SET2FORMULA.
Functions: COMPARISON, POL, POLND, REG, REGND, LOESS, SSPLINE.
Genstat Reference Manual 1 Summary sections on: Data structures, Analysis of variance, Regression analysis, REML analysis of linear mixed models.

FOURIER

Calculates cosine or Fourier transforms of real or complex series.

Option

`PRINT` = *string tokens*	What to print (`transforms`); default `*`

Parameters

`SERIES` = *variates*	Real part of each input series
`ISERIES` = *variates*	Imaginary part of each input series
`TRANSFORM` = *variates*	To save real part of each output series
`ITRANSFORM` = *variates*	To save imaginary part of each output series
`PERIODOGRAM` = *variates*	To save periodogram of each transform

Description

The Fourier or spectral analysis of time series is described comprehensively by Bloomfield (1976) and Jenkins & Watts (1968). The Fourier transformation of a series calculates the coefficients of the sinusoidal components into which the series can be analysed. There are four types of transformation described below, which are appropriate for different types of symmetry in the series. You may often want the length of the variate holding the supplied series to determine implicitly a natural grid of frequencies at which values of the transform are calculated. Genstat will do this if you have not previously declared the identifier supplied for the transform. Alternatively you may want to determine the transform at a finer grid of frequencies, and you can achieve this by declaring a transform variate that is as long as you require. You can do this only for the two types of Fourier transform that apply to real series.

Series of real numbers are stored in single variates, and series of complex numbers in pairs of variates. You can use the `FOURIER` directive to calculate the cosine transform of the real series $\{ a_t, t=0...N-1 \}$ stored in a variate `A` by

```
FOURIER [PRINT=transform] A
```

You calculate the Fourier transform of the complex series $\{ a_t+ib_t, t=0...N-1 \}$ by storing the values a_t in one variate, `A` say, the corresponding values b_t in another, `B` say, and giving the statement:

```
FOURIER [PRINT=transform] A; ISERIES=B
```

You can restrict the series specified by either the `SERIES` or `ISERIES` parameter to a contiguous set of units. Genstat then applies the transformation only to the restricted series of values. Similarly, you may supply restricted variates with the `TRANSFORM` and `ITRANSFORM` parameters to save the transform: Genstat will then carry out the transformation so as to supply the required number of values. There must be no missing values in the variates in the `SERIES` or `ISERIES` parameters, unless you exclude them by a restriction.

Genstat carries out the Fourier transformation using a fast algorithm which relies on the order of the transformation being highly composite (de Boor 1980). In practice, an appropriate order is a round number such as 300 or 6000, consisting of a digit followed by zeroes. If, however, the order has a large prime factor, the transformation may take much longer. For example, a transformation of order 499 is about 25 times slower than one of order 500. In the descriptions below, therefore, we clearly state the order of each form of the transformation, to illustrate a sensible choice of size.

The *cosine transformation of a real series* can be used to calculate the spectrum from a set of autocorrelations. Suppose the variate `R` contains the values $r_0 \ldots r_n$, and the variate `F` is to hold the calculated values $f_0 \ldots f_m$ of the spectrum. These values correspond to angular frequencies of $\pi j/m$; that is, periods of $2m/j$, for $j=0...m$. You apply the transformation by putting

```
FOURIER R; TRANSFORM=F
```

If F has not been declared previously, this statement defines it automatically as a variate with n+1 values (so m=n). If F has been declared to have m+1 values, then m must be greater than or equal to n; otherwise Genstat will redeclare F to have n+1 values.

The transform is defined when m>n by

$$f_j = r_0 + \sum_{k=1 \ldots n} \{ 2 \times r_k \times \cos(k \times \pi \times j / m) \}$$

When m=n the final term in this sum is

$$r_n \cos(\pi j) = r_n (-1)^j$$

and it appears without the multiplier 2. The order of the transformation is 2m.

If R contains sample autocorrelations, you must multiply it by a variate holding a lag window in order to obtain a smooth spectrum estimate (see Bloomfield 1976, page 166; or Jenkins & Watts 1968, page 243).

The *Fourier transformation of a real series* can be used to calculate the periodogram of a time series. Suppose the variate X of length N contains the supplied series values $x_0 ... x_{N-1}$. The result of the transformation is a set of coefficients $a_0 ... a_m$ of the cosine components and $b_0 ... b_m$ of the sine components of the series, held in variates A and B, say. Normally the number of such components is related to the length of the series by taking m=N/2 if N is even or m=(N−1)/2 if N is odd. Then the coefficients correspond to angular frequencies of 2πj/N, which is the same as saying that they correspond to periods N/j for j=0...m. Since by definition b_0=0, and b_m=0 if N is even, there are N "free" coefficients in A and B (which you can think of as the real and imaginary parts of a complex transform with values a_j+ib_j). You can save the periodogram values $p_0 ... p_m$ in a variate P, say: these are the squared amplitudes of the sinusoidal components, and are calculated by Genstat as $p_j = a_j^2 + b_j^2$.

You obtain the transform by putting

```
FOURIER X; TRANSFORM=A; ITRANSFORM=B; PERIODOGRAM=P
```

If you want only the periodogram, you can put

```
FOURIER X; PERIODOGRAM=P
```

If you have not declared A previously Genstat defines it automatically, here as a variate of length m+1 where m has the default value defined above. If you have previously declared A, it should have length greater than or equal to m+1; otherwise Genstat declares it to have this length. In any case, B and P should have the same length as A, and will be declared (or redeclared) if required.

In the usual case when A, B or P has the default length m+1, the transform is defined by:

$$a_j = \sum_{t=0 \ldots N-1} \{ x_t \times \cos(t \times 2\pi \times j / N) \} ; \quad j = 0 \ldots m$$

$$b_j = \sum_{t=0 \ldots N-1} \{ x_t \times \sin(t \times 2\pi \times j / N) \} ; \quad j = 0 \ldots m$$

In this case, the order of the transformation is N. If A, B and P have length m'+1 with m'>m, Genstat computes the results at a finer grid of frequencies 2πj/N', j=0...m' where N'=2m'. These replace 2πj/N in the above defining sums. The upper limit on the sums remains as N−1, although internally Genstat treats it as N'−1 with the extra values of $x_N ... x_{N'-1}$ being taken as zero. The order of the transformation is then N'. There are various conventions used for scaling the periodogram with factors 2/m, 1/m or 1/πm. You can apply these by using a CALCULATE statement after the transformation. You may also want to apply mean correction to the series before calculating the periodogram.

The *Fourier transformation of a complex series* is the most general form of the Fourier transformation; the other three types are essentially special cases in which some coefficients are zero or have a symmetric structure. Suppose variates X and Y contain values $x_0 ... x_{N-1}$ and $y_0 ... y_{N-1}$, which may be viewed as the real and imaginary parts of the series $\{ x_t + iy_t, t=0 ... N-1 \}$. The results of the transformation are coefficients $a_0 ... a_{N-1}$ and $b_0 ... b_{N-1}$ which can be held in variates A and B, say: these may similarly be considered as parts of complex coefficients a_t+ib_t, t=0 ... N−1.

You can do the transformation by putting

```
FOURIER SERIES=X; ISERIES=Y; TRANSFORM=A; ITRANSFORM=B
```

Both X and Y must be variates with the same length N. Similarly A and B must have length N, and if they do not Genstat will declare (or redeclare) them as variates of length N. The order of the transformation is N.

The results are defined by

$$a_j = \sum_{t=0\ldots N-1} \{ x_t \times \cos(t \times 2\pi \times j / N) - y_t \times \sin(t \times 2\pi \times j / N) \} \; ; j = 0 \ldots m$$
$$b_j = \sum_{t=0\ldots N-1} \{ x_t \times \sin(t \times 2\pi \times j / N) + y_t \times \cos(t \times 2\pi \times j / N) \} \; ; j = 0 \ldots m$$

or equivalently in complex form by

$$(a_j + i\, b_j) = \sum_{t=0\ldots N-1} \{ (x_t + i\, y_t) \times \exp(i\, t \times 2\pi \times j / N) \}$$

The complex transform can be used in cross-spectral analysis.

You can view a Fourier transformation as an orthogonal matrix transformation. Hence its inverse is another Fourier transformation (apart from some simple scaling). You can use this to calculate convolutions. In particular, the correlations of a time series can be obtained by applying the inverse cosine transformation to the periodogram.

The *Fourier transform of a conjugate sequence* is most easily considered as the reverse of the transformation of a real series, with the roles of the series and the transform interchanged. For the true inverse transformation some simple scaling is also required.

Thus if variates A and B of length m+1 are supplied containing values $a_0 \ldots a_m$ and $b_0 \ldots b_m$, which may be viewed as parts of complex coefficients a_j+ib_j, the result of the transformation is a single real series $x_0 \ldots x_{N-1}$ held in a variate X of length N.

X can be declared to have length N=2m or N=2m+1 (corresponding to the case N even or odd in the Fourier transformation of a real series). The value of b_0 must be zero; also if N=2m, the value of b_m must be zero. If either of these conditions is not satisfied, Genstat sets the values of these elements to zero and gives a warning. If X has not been declared previously (or has been declared with a length equal to neither 2m nor 2m+1), then it is declared (or redeclared) with a length governed by whether b_m is 0: N=2m if b_m=0, and N=2m+1 if $b_m \neq 0$. The value of b_0 is checked to be zero as before.

You can obtain the transform using the statement

```
FOURIER SERIES=A; ISERIES=B; TRANSFORM=X
```

The definition of the transform is, in the case N=2m+1,

$$x_t = a_0 + \sum_{j=1\ldots m} \{2 \times (a_j \times \cos(t \times 2\pi \times j / N) - b_j \times \sin(t \times 2\pi \times j / N)) \}$$

In the case N=2m, the final term in the sum is simply

$$a_m \cos(t\pi) = a_m (-1)^t$$

and it appears without the multiplier 2. The order of this transformation is N.

Option: PRINT.
Parameters: SERIES, ISERIES, TRANSFORM, ITRANSFORM, PERIODOGRAM.

Action with RESTRICT

You can restrict the series specified by either the SERIES or ISERIES parameter to a contiguous set of units. Genstat then applies the transformation only to the restricted series of values. Similarly, you may supply restricted variates with the TRANSFORM and ITRANSFORM parameters to save the transform: Genstat will then carry out the transformation so as to supply the required number of values.

References

Bloomfield, P. (1976). *Fourier Analysis of Time Series: an Introduction*. Wiley, New York.

de Boor, C. (1980). FFT as nested multiplication, with a twist. *SIAM Journal of Scientific and Statistical Computing*, **1**, 173-178.

Jenkins, G.M. & Watts, D.G. (1968). *Spectral Analysis and its Applications*. Holden-Day, San Francisco.

See also
Directive: CORRELATE.
Procedures: DFOURIER, MCROSSPECTRUM, PERIODTEST, REPPERIODOGRAM, SMOOTHSPECTRUM.
Genstat Reference Manual 1 Summary section on: Time series.

FPSEUDOFACTORS

Determines patterns of confounding and aliasing from design keys, and extends the treatment model to incorporate the necessary pseudo-factors.

Options

TREATMENTSTRUCTURE = *formula*	Treatment model for the design
BLOCKSTRUCTURE = *formula*	Block model for the design
FACTORIAL = *scalar*	Limit on the number of factors in each treatment term
LROWS = *factors* or *scalars*	Numbers of levels of factors, or factors, corresponding to the rows of the key matrices
LCOLUMNS = *factors* or *scalars*	Numbers of levels of factors, or factors, corresponding to the columns of the key matrices
NEWTREATMENTSTRUCTURE = *identifier*	Store the extended treatment model
PSEUDOFACTORS = *pointer*	Pseudo-factors required for the keys
NPSEUDOFACTORS = *scalar*	Number of pseudo-factors required for the keys
KEYPSEUDOFACTORS = *matrix*	Key to generate the pseudo-factors from the treatment factors
KEYCONTRASTS = *matrix*	Key partitioning the treatment terms into orthogonal sets of contrasts

Parameters

KEY = *matrices*	Design keys
KEYINVERSE = *matrices*	Store the inverses of the design keys
ALIASSETS = *variates*	Stores aliasing information about the orthogonal sets of treatment contrasts
RESOLUTION = *scalars*	Saves the resolution number of the design constructed by each key

Description

The FPSEUDOFACTORS directive examines a list of design keys, specified using the KEY parameter, and forms pseudo-factors to allow the ANOVA directive to cope with partial confounding or aliasing in the design generated by the keys. The factors corresponding to the rows of the keys are specified by the LROWS option, and those for the columns are specified by the LCOLUMNS option. If you merely want to save the inverses of the keys, using the KEYINVERSE parameter, you can specify scalars defining the numbers of levels of the factors instead of the factors themselves. If LROWS is not specified, FPSEUDOFACTORS will take the factors from the formula specified by the TREATMENTSTRUCTURE parameter, in the order that they occur there. Similarly, the BLOCKSTRUCTURE option can provide a default for LCOLUMNS.

FPSEUDOFACTORS assumes that the design is formed by generating a replicate using each design key. The BLOCKSTRUCTURE option defines the block structure within each replicate, so the full block structure would be Rep/(#BLOCKSTRUCTURE) where Rep is a factor for the replicates. The TREATMENTSTRUCTURE option specifies the treatment terms to be estimated using the design, and the FACTORIAL option allows a limit to be set on the number of factors in the terms that are generated as, for example, in the ANOVA directive. FPSEUDOFACTORS examines the keys to see whether any treatment terms are partially aliased or partially confounded. Provided the factors of each such term all have the same (prime) number of levels it can then extend the treatment formula, inserting pseudo-factors for these terms, so that the ANOVA directive can produce a correct analysis. The extended formula can be saved using the NEWTREATMENTSTRUCTURE option, and the NPSEUDOFACTORS option saves the number of pseudo-factors that are needed. The pseudo-factors themselves are represented by the elements

of a pointer specified by the PSEUDOFACTORS option, and the KEYPSEUDOFACTORS option can save the key matrix required to generate their values from the values of the treatment factors.

FPSEUDOFACTORS can also determine the aliasing relationships of treatment terms in fractional factorial designs. The KEYCONTRASTS option can save a design key that partitions the treatment terms into orthogonal sets of contrasts. (The matrix thus has a row for each set of contrasts, and a column for each treatment factor.) The ALIASSETS parameter saves a variate, for each design key, with length equal to the number of rows in the KEYCONTRASTS matrix. The variate stores integers indicating the alias group of each set of contrasts so, if two elements of the variate are equal, this indicates that the corresponding sets of contrasts are aliased in the replicate generated by the design key concerned. The RESOLUTION parameter saves the resolution number for the replicate generated by each design key. This is the minimum number of factors involved in any pair of aliased terms.

Options: TREATMENTSTRUCTURE, BLOCKSTRUCTURE, FACTORIAL, LROWS, LCOLUMNS, NEWTREATMENTSTRUCTURE, PSEUDOFACTORS, NPSEUDOFACTORS, KEYPSEUDOFACTORS, KEYCONTRASTS.
Parameters: KEY, KEYINVERSE, ALIASSETS, RESOLUTION.

See also
Directives: AFMINABERRATION, GENERATE, FKEY.
Procedures: AGFACTORIAL, AKEY, ARANDOMIZE, FACDIVIDE, FBASICCONTRASTS.
Genstat Reference Manual 1 Summary sections on: Design of experiments, Analysis of variance.

FRAME

Defines the positions and appearance of the plotting windows within the frame of a high-resolution graph.

Options

`GRID` = *string tokens*	Specifies grid lines (`xy`, `xz`, `yx`, `yz`, `zx`, `zy`)
`BOXFRAME` = *string tokens*	Whether to include a box enclosing the entire frame (`include`, `omit`)
`BACKGROUND` = *scalars* or *texts*	Specifies the colour to be used for the background of the whole frame (where allowed by the graphics device)
`RESET` = *string token*	Whether to reset the window definition to the default values (`yes`, `no`); default `no`

Parameters

`WINDOW` = *scalars*	Window numbers
`YLOWER` = *scalars*	Lower y device coordinate for each window
`YUPPER` = *scalars*	Upper y device coordinate for each window
`XLOWER` = *scalars*	Lower x device coordinate for each window
`XUPPER` = *scalars*	Upper x device coordinate for each window
`YMLOWER` = *scalars*	Size of bottom margin (for x-axis labels)
`YMUPPER` = *scalars*	Size of upper margin (for overall title)
`XMLOWER` = *scalars*	Size of left-hand margin (for y-axis labels)
`XMUPPER` = *scalars*	Size of right-hand margin
`BACKGROUND` = *scalars* or *texts*	Specifies the colour to be used for the background in each window (where allowed by the graphics device)
`BOX` = *string tokens*	Whether to include a box enclosing the plotted graphic (`include`, `omit`)
`BOXSURFACE` = *string tokens*	Box to include in a surface plot (`full`, `bounded`, `omit`)
`BOXKEY` = *string tokens*	Box to draw around key (`full`, `bounded`, `omit`)
`PENTITLE` = *scalars*	Pen to use to write the overall title
`PENKEY` = *scalars*	Pen to use for the key
`PENGRID` = *scalars*	Pen to use to draw the grid lines
`SCALING` = *string tokens*	How to scale the axis in each window (`xyequal`, `xzequal`, `yzequal`, `xyzequal`)
`TPOSITION` = *string tokens*	Position of title (`right`, `left`, `center`, `centre`)
`CINTERIOR` = *scalars* or *texts*	Specifies the colour to be used for the interior of each window (where allowed by the graphics device)
`CFRAME` = *scalars* or *texts*	Specifies the colour to be used for the frame of each window (where allowed by the graphics device)
`CTITLE` = *scalars* or *texts*	Specifies the colour to be used for the title bar of each window (where allowed by the graphics device)
`AXES` = *identifiers* or *pointers*	Additional oblique axes to include in each window
`SAVE` = *pointers*	Saves details of the current settings for the window concerned

Description

You can define up to 256 different windows in which to plot graphics. Each window is a rectangular area of the screen which is defined using *normalized device coordinates* (NDC). These can have a range from 0.0 to 1.4 in both Y and X directions, but the usable area depends on the orientation of the device (as defined by the `DEVICE` directive). With a landscape device, your should use only the range 0.0 to 1.0 for Y, while for a portrait device you should use only

0.0 to 1.0 in the X direction. The mapping from NDC to physical coordinates on the current output device is performed internally, so the window definitions are independent of the choice of device. The actual size of a particular window on different devices will vary according to their relative physical sizes. The NDC system used for window definition is also completely independent of the values of the data that are to be plotted.

When you use FRAME, any aspects of the windows that you do not specify explicitly retain the values that they had immediately before the FRAME statement. Alternatively, you can specify option RESET=yes to reset all these aspects to the default values, defined by Genstat at the start of each job.

To define a window, the upper and lower bounds are required in both y- and x-directions; thus defining both the position and the size of the window. For example

```
FRAME WINDOW=1; YLOWER=0.25; YUPPER=0.75;\
      XLOWER=0; XUPPER=0.5
```

defines window 1 to be a square of size 0.5, whose bottom left corner is at the point (0.0,0.25) and whose top right corner is (0.5,0.75). This does not define the exact size of a graph plotted in this window, as margins may be required for the annotation and titles (see below).

If you do not specify all four values in the FRAME statement, the existing values are retained. A check is then made on the validity of the window bounds. The settings of YLOWER and XLOWER must be strictly less than those of YUPPER and XUPPER respectively; also, none of the bounds can be outside the permitted range, which is [0.0,1.0] on most graphics devices. You cannot use * to reset a bound to the default value; if you try to do so, Genstat will produce an error diagnostic. (Instead you can specify option RESET=yes, as explained above.)

All the windows have a default size defined when you start Genstat. Window 1 is the default window used for plots by DGRAPH, DCONTOUR, and so on, and is set up to be a square of size 0.75. The default key window is window 2, which is a rectangle of height 0.25 and width 0.75 located immediately below window 1. Windows 3 and 4 are the unit square [0,1]×[0,1] and windows 5, 6, 7 and 8 are the top-left, top-right, bottom-left, and bottom-right quarters respectively of the unit square. Windows 9, 10, 11 and 12 also divide the frame into quarters, but they have the full width (0 to 1) in the x-direction and quarter of the width in the y-direction, working from the top (i.e. 0.75 to 1 for window 9) to the bottom (i.e. 0 to 0.25 for window 12) of the frame. The remaining windows, from 13 to 32, also default to the unit square. You can use FRAME to modify the size or position of any of these windows.

Usually, a margin is provided around each plot so that there is room for the axes to be drawn, along with labelling and titles as specified by the XAXIS or YAXIS directives. By default, the margin size is designed to allow sufficient room for annotation to be added using the standard character size, as defined by the SIZEMULTIPLIER or SMLABEL parameters of PEN. If you use XAXIS or YAXIS to control the plotting of axes explicitly you may wish to alter the size of the margins, either to increase the space used for the axes or, alternatively, to maximize the space available for the graph itself. For example, if you alter the size of the labelling, by explicitly defining the relevant axis pens, more space may be required for the axes; otherwise the labels may be clipped at the window bounds. The parameters YMLOWER, YMUPPER, XMLOWER and XMUPPER can be used to set the space (in NDC) for the bottom, top, left-hand and right-hand margins respectively, and have initial default settings of 0.10, 0.07, 0.12 and 0.05.

On most devices the background colours of the window may be modified by setting the BACKGROUND, CINTERIOR, CFRAME and CTITLE parameters. The BACKGROUND parameter can be used to define the colour for the whole background, while the other parameters define specific aspects (overriding any setting of BACKGROUND): CINTERIOR defines the colour of the interior of the plot (where the points are plotted), CFRAME defines the colour of the outer frame (outside the interior), and CTITLE is the colour of the title bar. The parameters can be set either to a text containing the name of one of Genstat's pre-defined colours, or to a scalar containing a number defining a colour using the RGB system; see the PEN directive and the RGB function for details.

Similarly, the BACKGROUND option can define the background colour for the whole frame (which may include areas outside any of the windows). The special colour setting 'match' can be used to apply the colour from the preceding parameter to the next one: CFRAME would inherit the colour from CINTERIOR, and CTITLE would inherit from from CFRAME. For example,

```
FRAME 1; CINTERIOR='white'; CFRAME='ivory'; CTITLE='match'
```

will specify colour white for the inside of the plot, and ivory to all the area outside this.

The PENTITLE and PENKEY options allow you to define the pens to be used to write the overall title and the key in each window; the initial default is to use pen −5 and −6 respectively. The TPOSITION parameter can be used to specify the position of the title in each window: either left-justified, right-justified or centred. The initial default is that it is centred.

The GRID option allows you to request grid lines to be drawn in particular directions and planes (for all the windows listed by the WINDOW parameter). For example the setting xy requests lines in the xy plane running from the x-axis (that is, parallel to the y-axis), and the setting yx requests lines in the xy plane running from the y-axis (that is, parallel to the x-axis); so you can set both of these to obtain box markings in the xy plane. The PENGRID parameter specifies the pen to be used for the grid lines in each window; the initial default is to use pen −4. You must use the RESET option if you want to restore these pen numbers to the initial defaults. (Genstat does not allow you to set negative pen numbers explicitly.) The BOX parameter allows you to put a box around the window in plots other than surface plots; the initial default is to include this. The box for a surface plot is controlled by the BOXSURFACE option, and can either be a full box enclosing the whole graph, or a bounded box enclosing just the surface; the initial default is that no box is drawn. The BOXKEY parameter can request that either a full or a bounded box be drawn around each key; the initial default is to omit the box. Finally, the BOXFRAME option controls whether or not a box is drawn around the entire frame; the initial default is to omit the box.

The SCALING parameter enables you to request that scaling of the x-, y- or z-axes should be equal in each window. For example, the xyequal setting ensures that the x- and y-axes are scaled identically, the setting xyzequal ensures that all the axes have the same scaling, and so on.

The AXES parameter allows you to specify the identifier of an oblique axis (defined by the AXIS directive) that should be included in a window. If you want to include several axes, you can specify a pointer containing the identifiers of the required axes.

The current FRAME settings for a particular window can be saved in a pointer supplied by the SAVE parameter. The elements of the pointer are labelled to identify the components.

Options: GRID, BOXFRAME, RESET.

Parameters: WINDOW, YLOWER, YUPPER, XLOWER, XUPPER, YMLOWER, YMUPPER, XMLOWER, XMUPPER, BACKGROUND, BOX, BOXSURFACE, BOXKEY, PENTITLE, PENKEY, PENGRID, TPOSITION, SCALING, CINTERIOR, CFRAME, CTITLE, AXES, SAVE.

See also

Directives: AXIS, XAXIS, YAXIS, ZAXIS.

Procedures: BANK, DHELP, FFRAME.

Genstat Reference Manual 1 Summary section on: Graphics.

FRENAME

Renames files.

No options

Parameters

OLD = *texts*	Name of each file to rename
NEW = *texts*	New name for each file
OVERWRITE = *string tokens*	Whether to overwrite any existing files (yes, no); default no

Description

FRENAME allows you to rename external files. The names of the original files are specified, in texts, but the OLD parameter. The new file names are specified by the NEW parameter. If no path is included in the file name, it is assumed to be in the current working directory (which can be defined by the WORKINGDIRECTORY option of the SET directory). If you need to define the path, remember that the character \ is the continuation symbol in Genstat. So this character needs to be duplicated in a string to avoid Genstat interpreting it as a continuation: for example

```
FRENAME 'Today.Dat'; NEW='D:\\April\\18.dat'
```

renames the file Today.dat to become the file 18.dat in the directory (or folder) D:\April. As a more convenient alternative, the PC version of Genstat allows you to use / instead: i.e. you could put

```
FRENAME 'Today.Dat'; NEW='D:/April/18.dat'
```

By default FRENAME gives a fault if there is already a file with the new name, but you can set parameter OVERWRITE=yes to overwrite it.

Options: none.
Parameters: OLD, NEW, OVERWRITE.

See also

Directives: FCOPY, FDELETE, CLOSE, ENQUIRE, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

FRQUANTILES

Forms regression quantiles.

Options

`Y` = *variate*	Response variate
`DESIGNMATRIX` = *matrix*	Design matrix for the regression model
`TOLERANCE` = *scalar*	Tolerance for the algorithm; default 10^{-12}

Parameters

`PRQUANTILE` = *scalars*	Values for which to perform the quantile regressions
`RESIDUALS` = *variates*	Parameter estimates from each quantile regression
`ESTIMATES` = *variates* or *matrices*	Estimates from each quantile regression, either a variate of estimates for a specific quantile or, if `PRQUANTILE` is set to a missing value, a matrix with a row of estimates for every cumulative probability value in the `CUMPROBABILITIES` variate
`XBARQUANTILES` = *variates*	When `PRQUANTILE` is set to a missing value, saves the sum of the mean of each design column multiplied by its regression quantile for all the quantile solutions
`CUMPROBABILITIES` = *variates*	When `PRQUANTILE` is set to a missing value, saves the cumulative probabilitiy values at which the estimated regression quantiles change
`EXIT` = *scalars*	Saves an exit code, with 0 to indicate success

Description

`FRQUANTILES` calculates regression quantile statistics using the algorithm of Koenker & D'Orey (1987). The `Y` option specifies the response variate, and the `DESIGNMATRIX` option specifies the design matrix for the regression model to be fitted. The design matrix can be formed, for example, using the `TERMS` directive.

The `PRQUANTILE` parameter can be set to a scalar specifying the probability value whose quantiles are required. The `ESTIMATES` parameter then saves the estimated regression quantile statistics, and the `RESIDUALS` parameter saves the corresponding residuals.

Alternatively, if `PRQUANTILE` parameter is set to a scalar containing a missing value, `FRQUANTILES` forms the complete set of "solutions" by finding all the probability values at which the regression quantiles change. These cumulative probabilities can be saved in a variate, using the `CUMPROBABILITIES` parameter, and the `ESTIMATES` parameter then saves a matrix with a row of estimates for each cumulative probability. The `XBARQUANTILES` parameter saves a variate containing the sum of the mean of each column of the `DESIGNMATRIX` multiplied by its regression quantile for all the cumulative probabilities.

The `EXIT` parameter can save a scalar containing an "exit" code, as follows:

0	the algorithm was successful;
1	the solution was not unique;
2	the algorithm failed.

If `EXIT` is set, no Genstat diagnostic is given if the algorithm fails, unless the failure arises from an incorrect option or parameter setting, or because Genstat has run out of workspace.

Options: `Y`, `DESIGNMATRIX`, `TOLERANCE`.
Parameters: `PRQUANTILE`, `RESIDUALS`, `ESTIMATES`, `CUMPROBABILITIES`, `XBARQUANTILES`, `EXIT`.

Method
For more details of quantile regression and of the estimation method, see Koenker (2005) and Koenker & D'Orey (1987).

Action with `RESTRICT`
`FRQUANTILES` takes account of restrictions on the `Y` variate.

References
Koenker, R. (2005). *Quantile Regression*. Cambridge University Press, New York.
Koenker, R.W. & D'Orey, V. (1987). Algorithm AS229 computing regression quantiles. *Applied Statistics*, **36**, 383-393.

See also
Directives: `FIT`, `TERMS`.
Procedures: `RQLINEAR`, `RQNONLINEAR`, `RQSMOOTH`.
Function: `RQOBJECTIVE`.
Genstat Reference Manual 1 Summary sections on: Regression analysis, Calculations and manipulation.

FSIMILARITY

Forms a similarity matrix or a between-group-elements similarity matrix or prints a similarity matrix.

Options

PRINT = *string token*	Printed output required (similarities, summary); default * i.e. no printing
STYLE = *string token*	Print percentage similarities in full or just the 10% digit (full, abbreviated); default full
METHOD = *string token*	Form similarity matrix or rectangular between-group-element similarity matrix (similarities, betweengroupsimilarities); default simi
SIMILARITY = *matrix* or *symmetric matrix*	Input or output matrix of similarities; default *
GROUPS = *factor*	Grouping of units into two groups for between-group-element similarity matrix; default *
PERMUTATION = *variate*	Permutation of units (possibly from HCLUSTER) for order in which units of the similarity matrix are printed; default *
UNITS = *text* or *variate*	Unit names to label the rows of the similarity matrix; default *
MINKOWSKI = *scalar*	Index *t* for use with TEST=minkowski

Parameters

DATA = *variates* or *factors*	The data values
TEST = *string tokens*	Test type, defining how each DATA variate or factor is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variate or factor
RANGE = *scalars*	Range of possible values of each DATA variate or factor; if omitted, the observed range is taken

Description

The FSIMILARITY directive forms similarity matrices, essentially using the method described by Gower (1971). The similarity coefficient that is calculated allows variables to be qualitative, quantitative or dichotomous, or mixtures of these types; values of some of the variables may be missing for some samples. The values of a similarity coefficient vary between zero and unity: two samples have a similarity of unity only when both have identical values for all variables; a value of zero occurs when the values for the two samples differ maximally for all variables.

You can form a symmetric matrix of similarities, or a rectangular matrix of similarities between the units in two groups. You can save either form of similarity matrix, using the SIMILARITY option. FSIMILARITY can also be used to print the symmetric matrix of similarities after it has formed it; alternatively, you can input an existing similarity matrix for printing, using the SIMILARITY option.

The DATA parameter specifies a list of variates or factors, all of which must be of the same length. If you want to print an existing similarity matrix, the DATA parameter (and the TEST and

RANGE parameters) should be omitted, and the SIMILARITY option used to input the matrix concerned.

The TEST parameter specifies a list of strings, one for each variate or factor in the DATA parameter list, that define their "types". If you want to exclude a variate or factor from contributing, you should specify an empty string (* or ''). Otherwise the similarity between units *i* and *j* is calculated as

$$\sum_k \{ w_k(x_{ik}, x_{jk})\, s_k(x_{ik}, x_{jk}) \} / \sum_k w_k(x_{ik}, x_{jk})$$

where x_{ik} is the value of the DATA variate *k* in unit *i*, and the contribution functions s_k and weight functions w_k for a variate or factor *k* of the available types are defined in the tables below (for further details see Gower 1971, 1985).

The first table contains the types appropriate for variates that are recording the presence or absence of a characteristic; these cannot be used with factors.

Type	Contribution s_k	Weight w_k
Jaccard	if $x_i \neq 0$ and $x_j \neq 0$, then 1	1
	if $x_i = x_j = 0$, then 0	0
	if only one of x_i or $x_j = 0$, then 0	1
RussellRao	if $x_i \neq 0$ and $x_j \neq 0$, then 1	1
	if $x_i = 0$ or $x_j = 0$, then 0	1
Dice	if $x_i \neq 0$ and $x_j \neq 0$, then 1	1
	if $x_i = x_j = 0$, then 0	0
	if only one of x_i or $x_j = 0$, then 0	0.5
antidice	if $x_i \neq 0$ and $x_j \neq 0$, then 1	1
	if $x_i = x_j = 0$, then 0	0
	if only one of x_i or $x_j = 0$, then 0	2
SneathSokal	if $x_i \neq 0$ and $x_j \neq 0$, then 1	1
	if $x_i = x_j = 0$, then 1	1
	if only one of x_i or $x_j = 0$, then 0	0.5
RogersTanimoto	if $x_i \neq 0$ and $x_j \neq 0$, then 1	1
	if $x_i = x_j = 0$, then 1	1
	if only one of x_i or $x_j = 0$, then 0	2

The simplematching type is appropriate for qualitative variables, which may be either variates or factors.

Type	Contribution s_k	Weight w_k
simplematching	if $x_i = x_j$, then 1	1
	if $x_i \neq x_j$, then 0	1

The next table shows the types that can be used for quantitative variates (but not factors). In the

definitions, r is the range of the variate, t is the Minkowski index (defined by the MINKOWSKI option). Note, however, that BrayCurtis and Soergel should not be mixed with other types.

Type	Contribution s_k	Weight w_k
cityblock	$1 - \|x_i - x_j\| / r$	1
Manhattan	synonymous with cityblock	
ecological	$1 - \|x_i - x_j\| / r$	1
	unless $x_i = x_j = 0$	0
Euclidean	$1 - \{(x_i - x_j) / r\}^2$	1
Pythagorean	synonymous with Euclidean	
Minkowski	$1 - \|x_i - x_j\|^t / r^t$	1
Divergence	$1 - \{(x_i - x_j) / (x_i + x_j)\}^2$	1
Canberra	$1 - \|x_i - x_j\| / (\|x_i\| + \|x_j\|)$	1
BrayCurtis	$1 - \|x_i - x_j\| / (x_i + x_j)$	$x_i + x_j$
Soergel	$1 - \|x_i - x_j\| / \max(x_i, x_j)$	$\max(x_i, x_j)$

The RANGE parameter contains a list of scalars, one for each variate or factor in the DATA list. This allows you to check that the values of each variate lie within the given range. If any variate or factor fails the range check, FSIMILARITY gives an error diagnostic and terminates without forming the similarity matrix. The range is also used to standardize quantitative variates; this allows you to impose a standard range, for example when variates are measured on commensurate scales. You can omit the RANGE parameter for all or any of the variates or factors by giving a missing identifier or a scalar with a missing value; Genstat then uses the observed range. If PRINT=summary, Genstat prints the name, the minimum value, and the range for each variate and factor.

The METHOD option controls what type of matrix is produced. METHOD=similarity, the default, gives a symmetric matrix of similarities amongst a single set of units. METHOD=betweengroupsimilarity gives a rectangular matrix of similarities between two sets of units. To form a rectangular matrix of similarities, you must also define the grouping of units by setting the GROUPS option (see below).

The PRINT, STYLE and PERMUTATION options govern the printing of a symmetric matrix of similarities. You can either form the similarity matrix within FSIMILARITY, or input it by the SIMILARITY option. To print the similarity matrix you should set option PRINT=similarity. The STYLE option has two settings, full (the default) or abbreviated. The similarity matrix printed in full style has its values displayed as percentages with one decimal place. If you put STYLE=abbreviated, the values of the similarity matrix are printed as single digits with no spaces, the digit being the 10's value of the similarity as a percentage. In both cases, though, the actual similarities in the range 0-1 are stored in the similarity matrix itself. The PERMUTATION option allows you to specify a variate with values corresponding to the order in which you want the rows of the similarity matrix to be printed. The reordering of the rows is most effective when the permutation arises from a hierarchical clustering and corresponds to the dendrogram order.

You use the GROUPS option to specify a partition of the units into two groups, by giving a factor with two levels. The units with level 1 of the factor correspond to the rows of the matrix, while the units with level 2 correspond to the columns.

The UNITS option allows you to label the rows of the output similarity matrix if the variates of the DATA parameter do not have any unit labels, or if you want to use different labels from those labelling the units of the variates. This labelling also applies to the rows and columns of a matrix of similarities between group elements.

Options: PRINT, STYLE, METHOD, SIMILARITY, GROUPS, PERMUTATION, UNITS, MINKOWSKI.
Parameters: DATA, TEST, RANGE.

Action with RESTRICT
If any of the DATA variates or factors is restricted, or if the factor in the GROUPS option is restricted, then that restriction is applied to all the variates or factors. If more than one is restricted, then the restrictions must all be to the same set of units. The dimension of the resulting symmetric matrix of similarities is taken from the number of units that contribute to the similarity matrix.

References
Gower, J.C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, **27**, 857-871.
Gower, J.C. (1985). Measures of similarity, dissimilarity and distance. In: *Encyclopedia of Statistical Sciences, Volume 5*, 397-405.

See also
Directives: CLUSTER, HCLUSTER, PCO, HREDUCE.
Procedures: ECANOSIM, HBOOTSTRAP, MANTEL, MASCLUSTER.
Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Multivariate and cluster analysis.

FSSPM

Forms the values of SSPM structures.

Options

PRINT = *string tokens*	Printed output required (correlations, wmeans, SSPM); default * i.e. no printing
WEIGHTS = *variate* or *symmetric matrix*	Variate of weights for weighted SSP, or symmetric matrix of weights (one row and column for each unit of data); default * i.e. all units with weight one
SEQUENTIAL = *scalar*	Used for sequential formation of SSPMs; a positive value indicates that formation is not yet complete (see READ directive); default * i.e. not sequential

Parameter

SSPMs	Structures to be formed

Description

FSSPM forms the values for the component parts of SSPM structures, based on the information supplied when the SSPM directive was used to declare them. You can use an SSPM as input to the regression directive TERMS, or the multivariate directives PCP and CVA. The method used to form the SSPM is based on the updating formula for the means and corresponding corrected sums of squares and cross products (Herraman 1968).

FSSPM has one parameter which lists the SSPM structures whose values are to be formed. If any of these vectors has a missing value, the corresponding unit is excluded from all the means and all the sums of squares and products. You can also exclude units by setting their weights to zero.

When you have very many units, you may not be able to store them all at the same time within Genstat. You can then use the SEQUENTIAL option of READ to read the data in conveniently sized blocks, and the SEQUENTIAL option of FSSPM to control the accumulation of the sums of squares and products. The SSPM is updated for each block of data in turn until the end of data is found. The PRINT option has no effect until the last set of values is processed, when READ sets the scalar indicator to a negative value.

Options: PRINT, WEIGHTS, SEQUENTIAL.
Parameter: unnamed.

Action with RESTRICT

FSSTM takes account of restrictions on any of the variates or factors forming the terms of the SSPM, or on the weights variate or grouping factor if you have specified them.

Reference

Herraman, C. (1968). Algorithm AS12: Sums of squares and products matrix. *Applied Statistics*, **17**, 289-292.

See also

Directives: SSPM, CVA, FCA, PCO, PCP, TERMS.
Procedures: FCORRELATION, FVCOVARIANCE, ROBSSPM.
Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Multivariate and cluster analysis, Regression analysis.

FTSM

Forms preliminary estimates of parameters in time-series models.

Option

PRINT = *string tokens*	What to print (models); default *

Parameters

TSM = *TSMs*	Models whose parameters are to be estimated
CORRELATIONS = *variates*	Auto- or cross-correlations on which to base estimates for each model
BOXCOXTRANSFORM = *scalars*	Box-Cox transformation parameter
CONSTANTTERM = *scalars*	Constant term
VARIANCE = *scalars*	Variance of ARIMA model, or ratio of input variance to output variance for transfer model

Description

The TFIT directive carries out a lot of computation to find the best estimates of the parameters of a time-series model. The amount of computation can be reduced if you provide rough initial values for the parameters, especially when there are many of them. This can be done using the FTSM directive. FTSM obtains moment estimators of a simple kind, by solving equations between the unknown parameters of the ARIMA or transfer-function model and the autocorrelations or cross-correlations calculated from the observed time series. Sometimes these equations have no solution, or their solution provides values inconsistent with the constraints demanded of the parameters. If so, Genstat sets the corresponding parameters to missing values. The form of the directive is the same for ARIMA and transfer-function models, but the interpretation is slightly different.

To obtain preliminary estimates of ARIMA model parameters, a typical FTSM statement might be

```
FTSM [PRINT=model] Yatsm; CORRELATIONS=Yacf; BOXCOX=Ytran;\
  CONSTANT=Ymean; VARIANCE=Yvar
```

You must previously have declared Yatsm to be a TSM structure (i.e. a time-series model structure) of type ARIMA with appropriate orders, and lags if you need to specify them. Genstat takes this model to be associated with observations of a time series y_t. The aim of the directive is to set the values of the variate of model parameters equal to preliminary estimates derived from the variate Yacf and scalars Ytran, Ymn and Yvar.

The variate Yacf should contain sample autocorrelations r_0 ... r_m. You should obtain these from the original time series, stored in variate Y say, by first using the CALCULATE directive to transform Y according to the Box-Cox equations with transformation parameter Ytran (if you do indeed want a transformation). You should then form the differences of the transformed series, according to the degrees of differencing already set in the model; you can use the DIFFERENCE function with the CALCULATE directive for this. Finally, you should use the AUTOCORRELATIONS parameter of the CORRELATE directive to store the autocorrelations of the resulting series in Yacf. Often you will have done these operations already in order to produce Yacf for selecting a model.

At the same time, you can supply the scalars Ytran, Ymean and Yvar to set the first three elements of the parameters variate of Yatsm; these cannot be set using Yacf alone. The scalar Ytran should be the parameter used to transform Y, and Genstat will copy it into the first element of the variate of parameters. Genstat will copy the scalar Ymean into the second element, which is the constant term of the model; the recommended value for this is the sample mean of the series from which Yacf is calculated, but you may prefer the value 0. The scalar Yvar is used to set the innovation variance, which is the third element of the variate of parameters. The

recommended value is the sample variance of the series from which `Yacf` is calculated. If you set `Yvar` to 1.0, then Genstat will set the innovation variance to the variance ratio Variance(e)/Variance(y), as estimated from `Yacf` according to the model.

If any of the `BOXCOX`, `CONSTANT` or `VARIANCE` parameters is not set, Genstat will leave unchanged the corresponding value in the variate of parameters of the model. The only exception to this rule is if a parameter is missing. Then Genstat initially sets the transformation parameter to 1.0 (corresponding to no transformation), and the constant to 0.0; the innovation variance is left missing.

A typical `FTSM` statement for a transfer-function model might be

```
FTSM [PRINT=model] Xytsm; CORRELATIONS=Xyccf; BOXCOX=Xtran;\
  CONSTANT=Xmean; VARIANCE=Xyvratio
```

You must previously have declared the time-series model `Xytsm` to be of type `transferfunction` with appropriate orders, and lags if you need to specify them. Genstat assumes that this model represents the dependence of an output series y_t on an input series x_t in a multi-input model. The directive sets the values of the parameters of the model equal to preliminary estimates derived from `Xyccf`, `Xtran`, `Xmean` and `Xyvratio`.

You should put into the variate `Xyccf` an estimate of the impulse-response function of the model, from which Genstat will derive the parameters. This estimate is usually a sample cross-correlation sequence $r_0 \ldots r_m$ obtained from variates `Y` and `X1` containing observations of y_t and x_t according to one of the following four rules:

(a) In the simple case, the differencing orders of `Xytsm` are all zero, and you do not want to use any Box-Cox transformation of either y_t or x_t. Then the cross-correlations should be those between variates `Alpha` and `Beta`, say, derived from `X` and `Y` by filtering (or pre-whitening), see the `TFILTER` directive. The ARIMA model that you used for the filter should be the same for `X` and `Y`, and you should choose it so that the values of `Alpha` represent white noise.

(b) If the differencing orders of `Xytsm` are not zero, then before you calculate the cross-correlations you should further difference the series `Beta` as specified by these orders.

(c) If a Box-Cox transformation is associated with y_t, you should apply it to `Y` before the filtering. However this transformation parameter must not be associated with `Xytsm`: you should assign it to the univariate ARIMA model that you have specified for the error term.

(d) If a Box-Cox transformation is associated with x_t, it must be the same as the one you used in the ARIMA model for x_t from which the series `Alpha` was derived. The scalar `Xtran` must contain this transformation parameter. Genstat copies it into the first element of the parameter variate of `Xytsm`. If the Box-Cox parameter is unset, Genstat leaves the transformation parameter of `Xytsm` unchanged; it is set to 1.0 if it was originally missing.

Genstat copies the scalar `Xmean` into the second element of the variate of parameters. The recommended value is the sample mean of `X` after any transformation has been applied. If you do not set the `CONSTANT` parameter, Genstat leaves the constant parameter of `Xytsm` unchanged; it is set to 0.0 if it was originally missing.

You use the scalar `Xyvratio` to obtain the correct scaling of non-seasonal moving-average parameters in `Xytsm`. All the other autoregressive parameters and moving-average parameters are invariant under scale changes in y_t and x_t. You should set the scalar to the ratio of the sample variances of the variates from which the cross-correlations were calculated; that is, Variance(`Beta`)/Variance(`Alpha`). If you do not set this, Genstat uses the value 1.0.

You can use `FTSM` to go backwards from autocorrelations to the original time-series model. If you apply it to the autocorrelations that were constructed from a time-series model by means of `TSUMMARIZE`, it will recover the parameters of the model exactly, provided the model is non-seasonal. If the model contains seasonal parameters, with seasonal period s, the parameters will not be recovered exactly, except in one special circumstance: that is, when the non-seasonal part of the model, considered in isolation from the seasonal part, has a theoretical autocorrelation

function that is zero beyond lag $s/2$. Otherwise, the non-seasonal and seasonal parts of the model interact, and so Genstat loses accuracy in the recovered parameters. When you use sample autocorrelations, this loss of accuracy tends to be small in comparison with the sampling fluctuations of the estimates. But if s is small, say s=4 for quarterly data, the loss could be serious. Exactly the same considerations apply to transfer-function models.

Option: PRINT.
Parameters: TSM, CORRELATIONS, BOXCOXTRANSFORM, CONSTANT, VARIANCE.

See also
Directives: TSM, TDISPLAY, TFILTER, TFIT, TFORECAST, TKEEP, TRANSFERFUNCTION, TSUMMARIZE.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

FVARIOGRAM

Forms experimental variograms.

Options

PRINT = *string token*	Controls printed output (statistics); default stat
Y = *variate*	Y positions (needed only for 2-dimensional irregular data)
X = *variate*	X positions or interval (not needed for 2-dimensional regular data i.e. when DATA is a matrix)
YMAX = *scalar*	Maximum lag in the y direction (2-dimensional regular data only)
XMAX = *scalar*	Maximum lag in the x direction
STEPLENGTH = *scalar* or *variate*	Length(s) of the steps in which lag is incremented
METHOD = *string token*	How to estimate the variogram (moments, cressiehawkins, dowd, genton); default mome
DIRECTIONS = *scalar* or *variate*	Directions (degrees) along which to form the variogram (relevant only for 2-dimensional irregular data)
SEGMENTS = *scalar* or *variate*	Angles subtended by the segments (degrees) over which averaging is to be done (relevant only for 2-dimensional irregular data)

Parameters

DATA = *variates* or *matrices*	Measurements as a variate or, for data on a regular grid, as a matrix
VARIOGRAMS = *variates* or *matrices*	Structure to store the sample variogram
COUNTS = *variates* or *matrices*	Numbers of comparisons involved in the calculation of each variogram
DISTANCES = *variates* or *matrices*	Mean lag distances at each step
LAGPOINTS = *pointer*	Saves lag classes, indexes to observations and directions to plot in an h-scattergram

Description

The FVARIOGRAM directive forms an experimental variogram from a set of values of a variable, *Z*, distributed in one or two dimensions. By default the variogram is calculated by Matheron's method of moments, as

$$\gamma(\mathbf{h}) = (1 / (2 \times m(\mathbf{h}))) \times \sum_{i=1 \ldots m(\mathbf{h})} \{ z(\mathbf{x}_i) - z(\mathbf{x}_i + \mathbf{h}) \}^2 ,$$

where $z(\mathbf{x}_i)$ and $z(\mathbf{x}_i + \mathbf{h})$ are the values at positions $\mathbf{x}_i + \mathbf{h}$, and $m(\mathbf{h})$ is the number of paired comparisons contributing to the estimate. For data on a regular grid or transect $\mathbf{h}$ is an integer multiple of the sampling interval. For irregularly scattered data $\mathbf{h}$ is discretized so that for each nominal lag there is a range of distance equal to the increment and an angular range set by the user. The nominal lag is at the centre of both ranges. However, you can set the METHOD option to calculate robust estimates instead. The cressiehawkins setting uses the estimator of Cressie & Hawkins (1980), which essentially damps the effect of outliers from the secondary process. Dowd's (1984) and Genton (1978) methods, which estimate the variogram for a dominant intrinsic process in the presence of outliers, can be requested by the dowd and genton settings respectively. For further details see Webster & Oliver (2007) pages 67-68 and 115-116.

The data are specified using the DATA parameter. If they are on a regular grid, they should be supplied in a matrix defined with a variate of column labels to provide the x-values and a variate of row labels to provide the y-values. Alternatively, if they are irregularly scattered, then they

should be supplied in a variate, and the X and Y options should be set to variates to supply their spatial coordinates.

The experimental variogram is controlled by five options. For irregular data the maximum distance to which the variogram is calculated is set by the XMAX option for all directions. For regular data XMAX defines the maximum lag distance in the X direction, and YMAX must also be given to limit the distance in the Y direction. The increments in distance are set by the STEPLENGTH option, where you can supply a scalar to define equally-spaced steps or a variate to specify the steps themselves. The variogram may be computed in one or more directions. These are given by the DIRECTIONS option in degrees counterclockwise from east in the usual convention. Each direction is at the centre of an angular range, which is defined by the SEGMENTS option. DIRECTIONS and SEGMENTS should be set to scalars if the variogram is to be calculated for only one direction, or to variates if there are to be several.

A variogram can be computed without regard to direction by setting DIRECTIONS to 0 and SEGMENTS to 180. This is advisable if variation seems to be isotropic, i.e. the same in all directions, or if there are too few data to compute $\hat{\gamma}(\mathbf{h})$ for two or more directions separately. The lag then becomes a scalar $|\mathbf{h}| = h$ in distance only. Experience suggests that some 300 data are needed to distinguish anisotropy.

By default some statistics are printed concerning the variogram, but these can be supressed by setting option PRINT=*. Other information can be saved using the various parameters, in variates if there is a single direction, or in matrices with one column for each direction if there are several: VARIOGRAMS stores the ordered set of semivariances; DISTANCES stores the mean lag distances at which the semivariances have been computed; and COUNTS stores the numbers of paired comparisons from which the semivariances have been computed.

The LAGPOINTS parameter allows you to save a pointer containing lag classes, indexes to observations and directions that can be used to plot an h-scattergram.

Options: PRINT, Y, X, YMAX, XMAX, STEPLENGTH, METHOD, DIRECTIONS, SEGMENTS.
Parameters: DATA, VARIOGRAMS, COUNTS, DISTANCES, LAGPOINTS.

Action with RESTRICT

You can restrict a DATA variate to form the variogram from only a subset of its units.

References

Cressie, N. & Hawkins, D.M. (1980). Robust estimation of the variogram. *Journal of the International Association of Mathematical Geology*, **12**, 115-125.

Dowd, P.A. (1984). The variogram and kriging: robust and resistant estimators. In: *Geostatistics for Natural Resources Characterization* (ed. G. Verly, M. David, A.G. Journel & A. Marechal), 91-106. D. Reidel, Dordrecht.

Genton, M.G. (1998). Highly robust variogram estimation. *Mathematical Geology*, **30**, 213-221.

Webster, R. & Oliver, M.A. (2007). *Geostatistics for Environmental Scientists, 2nd Edition*. Wiley, Chichester.

See also

Directives: KRIGE, FCOVARIOGRAM, MCOVARIOGRAM, COKRIGE.
Procedures: MVARIOGRAM, DVARIOGRAM, DCOVARIOGRAM, DHSCATTERGRAM, KCROSSVALIDATION.
Genstat Reference Manual 1 Summary section on: Spatial statistics.

GENERATE

Generates factor values for designed experiments.

Options

TREATMENTS = *formula*	Model term for which pseudo-factors are to be generated; default *
REPLICATES = *formula*	Factors defining replicates of the design; default *
BLOCKS = *formula*	Block formula (for design-key generation) or term (for generation of pseudo-factors); default *
KEY = *matrix*	Key matrix (number of factors in the parameter list by number of factors in the BLOCKS formula) to generate the factors by the design key method; default *
BASEVECTOR = *variate*	Base vector for design key generation; default *

Parameter

factors	Factors whose values are to be generated

Description

GENERATE is invaluable when you have a set of data that is to be read in a systematic order: for example, you may want to take all the observations within one group, then the same number of observations within the next group, and so on until an equal number of observations has been read for every group. You can then define values of the grouping factor or factors by GENERATE; so the only values that you need to read are the observed data. Designed experiments are the obvious instance where the data are structured in this way: for example, you might have all the data from the first block, then all those from the second block, and so on.

The best way to understand GENERATE is to look at some examples. The values of a set of factors that you have defined by GENERATE are said to be in *standard order*: that is their units are arranged so that the levels of the first factor occur in the same order as in its levels vector then, within each level of the first factor, the levels of the second factor are arranged similarly, and so on. For example

```
FACTOR [NVALUES=24; LEVELS=2] A
& [LEVELS=!(4,1,2)] B
& [LEVELS=4] C
GENERATE A,B,C
```

gives A, B and C the values

```
A: 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
B: 4 4 4 4 1 1 1 1 2 2 2 2 4 4 4 4 1 1 1 1 2 2 2 2
C: 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

Placing a number or a scalar in the parameter list has the same effect as if a factor with that number of levels had been listed. Thus to generate values only for A and C, all that you require is

```
GENERATE A,3,C
```

To generate values for just B and C is even simpler since the cycling process is itself recycled until all the units have been covered. Omitting A therefore causes all combinations of a level of B with a level of C to be used twice, in the same pattern as displayed above; so you need specify only

```
GENERATE B,C
```

You get a warning if one of the cycles is incomplete, as would happen for example if B and C had 18 values instead of 24.

This first use of GENERATE, then, is particularly appropriate for generating the blocking factors in an experimental design.

Another use, obtained by setting the BLOCKS, KEY and BASEVECTOR options, is to form values of treatment factors using the design-key method. This method, described by Patterson (1976) and Patterson & Bailey (1978), provides a very flexible way of specifying the allocation of treatments in an experimental design. The method assumes that the units are identified by a set of what are called "plot" factors. In Genstat terms, these will often be the same as the factors that occur in the block formula of the design (see the BLOCKSTRUCTURE directive), and they are specified by the BLOCKS option of GENERATE. The setting is a formula, but remember this can be just a list of factors if you do not wish to indicate their inter-relationships; if the setting is more than just a list, Genstat forms the set of plot factors by taking the factors from the block formula in the order in which they occur there. Of course, the factors need not be identical to those in the block formula. For example if one these factors has a non-prime number of levels, it may need to be specified instead as the combination of two or more (pseudo) factors: for example, in a block design with blocks of size eight, the plots might need to be indexed by three factors with two levels.

The treatment factors to be generated are again specified by the parameter of GENERATE.

The KEY option specifies a matrix known as the *design key*, which indicates how the values of each treatment factor are to be calculated from the plot factors. The matrix has a row for each treatment factor and a column for each plot factor; below k_{ij} represents the element in row i and column j. (This is the transpose of the form used by Patterson 1976, but in Genstat it seems more convenient to specify the treatments by rows.) There is also an option called BASEVECTOR, which can specify a variate with an element b_i for each treatment factor to allow the levels of the factor to be shifted cyclically; if this is unset, Genstat assumes b_i=0.

The calculation assumes that the values of the plot factors are represented by the integers zero upwards (and GENERATE will perform this mapping automatically if necessary). The value $q[i]_u$ in unit u of treatment factor i is then given by

$$q[i]_u = b_i + k_{i1} \times p[1]_u + k_{i2} \times p[2]_u + \ldots + k_{in} \times p[n]_u \quad \text{modulo } t_i$$

where $p[1]_u \ldots p[n]_u$ are the values of the plot factors in unit u, and t_i is the number of levels of treatment factor i. The calculated values are integers in the range 0, 1 ... t_i−1, but GENERATE will again map these to the defined levels if necessary.

To illustrate the process, the treatments to be allocated (before randomization) to the plots of an $n \times n$ Latin Square may be calculated as

$$\text{Latin-factor-value} = \text{Row-factor-value} + \text{Column-factor-value} \quad \text{modulo } n$$

The values of the extra factor in a Graeco-Latin square can then be formed as

$$\text{Graeco-factor-value} = \text{Row-factor-value} + 2 \times \text{Column-factor-value} \quad \text{modulo } n$$

So design key has rows (1,1) and (1,2).

The design key thus provides a very convenient way of defining treatment factors. Essentially, the key identifies each factor i with the set of contrasts (in the usual terminology)

P[1]**K_{i1} P[2]**K_{i2} ... P[n]**K_{in}

and the skill when forming a design is in selecting the best set for each factor. Further keys are presented by Patterson & Bailey (1978), and these are used in the example of procedure AKEY; this procedure extends the GENERATE facilities by allowing the block factors to be generated automatically, and the design to be printed after the factors have been generated. The Genstat design system has a repertoire of keys, used by procedures DESIGN and AGDESIGN to generate a range of designs including factorials, fractional factorials, Latin squares and Lattices. You can form your own keys for designs not covered by the repertoire, using the FKEY directive.

GENERATE can also be used to form the values of pseudo-factors in partially balanced designs. The treatment term to which the pseudo-factors are to be linked is specified by the TREATMENTS option. The factors that identify the replicates are specified by the REPLICATES option, and those that identify the blocks within each replicate are specified by the BLOCKS option. The settings of these two options are model formulae, but Genstat merely scans them to find which

factors they contain; so you may again find it easiest simply to give the factors as a list. The parameter of GENERATE lists the pseudo-factors. These have as many levels as there are blocks within each replicate. The blocks in the first replicate are used to determine which combinations of the factors in the treatment term correspond to each level of the first pseudo-factor, those in the second replicate are used for the second pseudo-factor, and so on. If a treatment combination occurs in more than one block within the same replicate, the level of the corresponding pseudo-factor is not determined uniquely and Genstat will report an error.

Options: TREATMENTS, REPLICATES, BLOCKS, KEY, BASEVECTOR.
Parameter: unnamed.

Action with RESTRICT
Any of the factors may be restricted to generate values for only a subset of the units.

References
Patterson, H.D. (1976). Generation of factorial designs. *Journal of the Royal Statistical Society, Series B*, **38**, 175-179.
Patterson, H.D. & Bailey, R.A. (1978). Design keys for factorial experiments. *Applied Statistics*, **27**, 335-343.

See also
Directives: FKEY, FPSEUDOFACTORS.
Procedures: AKEY, ARANDOMIZE, FACDIVIDE, FACPRODUCT, FBASICCONTRASTS.
Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Design of experiments,

GET

Accesses details of the "environment" of a Genstat job.

Options

†ENVIRONMENT = *pointer*	Pointer given unit labels 'inprint', 'outprint', 'diagnostic', 'errors', 'pause', 'prompt', 'newline', 'case', 'run', 'wordlength', 'captions', 'typeset', 'cmethod', 'dataspace', 'algorithms', 'actionafterfault', 'unsetdummy', 'language', 'year2digitbreak' and 'timewithseconds' to save the current settings of those options of SET; default *
SPECIAL = *pointer*	Pointer given unit labels 'units', 'blockstructure', 'treatmentstructure', 'covariate', 'asave', 'dsave', 'msave', 'rsave', 'tsave', 'vsave' and 'vcomponents, used to save the current settings of those options of SET; default *
LAST = *text*	To save the last input statement; default *
FAULT = *text*	To save the last fault code; default *
FIELDWIDTH = *scalar*	Saves the fieldwidth currently defined as the default minimum for PRINT and other output commands
SIGNIFICANTFIGURES = *scalar*	Saves the minimum number of significant figures currently to be supplied in the default formats determined by PRINT and other output commands
SEEDS = *pointer*	Saves a pointer to variates defining the seeds currently used as defaults by random-number functions, the RANDOMIZE directive, and internally by various other directives
EPS = *scalar*	To obtain the value of the smallest x (on this computer) such that $1+x > 1$; default *
NJOB = *scalar*	Number of the current job within the program; default *
VERSION = *pointer*	Information about the version of Genstat that is being used; default *
PID = *scalar*	Gets an integer value unique in the current job to use, for example, in names of temporary files
WORKINGDIRECTORY = *text*	Saves the name of the current working directory

No parameters

Description

The GET directive allows you to access the current settings of the environment. This can be particularly useful in procedures, when details of the environment may need to change and be reset later to their original state. Sometimes it may be sufficient just to use the RESTORE option of the PROCEDURE directive for this purpose, but this causes them to be reset only at the end of a procedure.

The ENVIRONMENT and SPECIAL options of GET are used to access and save the current settings of options of the SET directive. The options of SET are divided into two groups. Those that apply to the general environment can be saved using the ENVIRONMENT option: these are INPRINT, OUTPRINT, DIAGNOSTIC, ERRORS, PAUSE, PROMPT, NEWLINE, CASE, RUN,

WORDLENGTH, CAPTIONS, TYPESET, CMETHOD, DATASPACE, ALGORITHMS, ACTIONAFTERFAULT, UNSETDUMMY, LANGUAGE, YEAR2DIGITBREAK and TIMEWITHSECONDS. Those that apply only to the save structures associated with particular directives can be saved using the SPECIAL option: these are UNITS, BLOCKSTRUCTURE, TREATMENTSTRUCTURE, COVARIATE, ASAVE, DSAVE, MSAVE, RSAVE, TSAVE, VSAVE and VSTRUCTURE. The labels of the pointer can be specified in either lower or upper case. or any mixture.

When you use the ENVIRONMENT option, Genstat sets up a pointer structure with units identified by the labels of the corresponding options of SET: these labels are 'inprint', 'outprint', and so on. The labels can be specified in either lower or upper case, or any mixture. Each unit of this pointer contains one or more strings, or a scalar, to represent the current setting. Thus, the statement

```
GET [ENVIRONMENT=Env]
```

would set up a pointer called Env with elements Env['inprint'], Env['outprint'], and so on. Each element can also be referred to by its position in the pointer; for example, Env['inprint'] is the same as Env[1].

Thus you do not have to know how the environment has been set in order to change it and then restore it; you can use GET to find out about it, and SET to change it back. For example, suppose that you wanted to stop temporarily the echoing of statements to the output file in a batch program. In the following program the first SET statement cancels the echoing, if indeed any echoing is in progress, and the second restores echoing to what it was before the first SET.

```
GET [ENVIRONMENT=Env]
SET [INPRINT=*]
(more statements)
SET [INPRINT=#Env['inprint']]
```

The SPECIAL option similarly sets up a pointer to save its information. The labels of the pointer are 'units', 'blockstructure', and so on. These can again be specified in either lower or upper case, or any mixture. The first element of the pointer is the units structure, or, failing that, the number of units if you have defined it for the current job. Printing the contents of the other elements is not usually informative, as the information is stored in coded form. The last ten elements of the pointer allow you to access the special save structures in the graphical and analysis directives. They are most useful for recovering information about an analysis when you have not already specified an explicit save structure. (Otherwise you would have to do the analysis all over again.) The SPECIAL option also provides a way of accessing the save structures associated with the analysis-of-variance directives BLOCKSTRUCTURE, COVARIATE and TREATMENTSTRUCTURE. This facility is used by the ASTATUS procedure, which may be a more convenient way of accessing these structures.

The LAST option is used to save the latest statement that you have input. You can then give the statement again later in the job without having to retype it, though some implementations of Genstat provide a simpler recall facility using the cursor keys. The option has the same effect as setting up a macro containing a single statement, and is accessed in the same way. For example, the statements

```
PRINT [SERIAL=yes; IPRINT=*; SQUASH=yes] !t('New Data'),Y
GET [LAST=Prdat]
(statements)
READ Y
(data)
##Prdat
```

would print the data, Y, under the title New Data and save the PRINT statement in a text called Prdat. After the next data set is read, the heading New Data and the new data set are printed

in the same format as the previous data set.

The FAULT option is used to save the last fault code as a single string in a text structure. (A list of fault code definitions is available in the on-line help provided with most implementations of Genstat.) This option is particularly useful in procedures, in combination with the DIAGNOSTIC and FAULT options of SET, to control the printing of diagnostics.

The FIELDWIDTH option saves the fieldwidth currently defined as the default minimum for PRINT and other output commands, and the SIGNIFICANTFIGURES option saves the minimum number of significant figures currently to be supplied in the default formats determined by PRINT and other output commands (see PRINT for details).

The SEEDS option saves a pointer containing variates, each containing four values, which define the seeds currently used as defaults by random-number functions, the RANDOMIZE directive, and internally by various other directives. The pointer elements are labelled to identify the use of the seeds concerned: for example 'calculate', and 'randomize' for random-number functions and the RANDOMIZE directive respectively.

The EPS option is used to obtain the smallest number, ε, such that 1.0+ε is recognized by your computer to be greater than 1.0; this is an indication of the precision of the computer, which can affect the behaviour of some of the algorithms used by Genstat. EPS can be used, for example, when testing for convergence of iterative algorithms.

The NJOB option provides the current job number within the Genstat program. It is used in the start-up file to distinguish between statements to be executed just at the start of the program, and those to be executed at the start of each job.

The VERSION option provides information about the version of Genstat that is being used. This is particularly useful within general programs or procedures. It saves a pointer containing the following elements.

release	is a scalar storing the release number, for example 21.10. The main information is in the integer part and the first decimal place; the second decimal may be used to distinguish between sub-releases with minor changes or corrections.
patch	shows whether the release includes a patch.
build	is the build number (useful for support).
implementation	identifies the type of computer for which the version has been implemented, for example 'PC'.
system	indicates the operating system, for example Windows 11.
version	may contain further information relevant to particular implementations.
description	gives the name of the release, for example Genstat Twenty-first Edition.
bits	gives the number of bits for which the implementation has been built, for example 64 for a 64-bit version.

The PID option saves a scalar containing an integer value that is unique within the current job. You might want to use this, for example, to define a unique name for a temporary file.

The WORKINGDIRECTORY option saves a text containing the name of the current working directory.

Options: ENVIRONMENT, SPECIAL, LAST, FAULT, FIELDWIDTH, SIGNIFICANTFIGURES, SEEDS, EPS, NJOB, VERSION, PID, WORKINGDIRECTORY.
Parameters: none.

See also
Directives: SET, PROCEDURE.
Genstat Reference Manual 1 Summary section on: Program control.

GETLOCATIONS

Finds locations of an identifier within a pointer, or a string within a factor or text, or a number within any numerical data structure.

Options

CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for a string (significant, ignored); default sign
TOLERANCE = *scalar*	Tolerance for comparing numbers
SUBSTITUTE = *string token*	Whether to substitute dummies within pointers in DATA or FIND (yes, no); default no

Parameters

DATA = *identifiers*	Variates, scalars, matrices, tables, factors, texts or pointers to be searched
FIND = *scalars*, *texts* or *pointers*	Numbers, strings or identifiers to be located in DATA
NLOCATIONS = *scalars*	Saves the number of times that FIND occurs in DATA
LOCATIONS = *variates* or *pointers*	Saves the locations where FIND occurs as one of the values in DATA, in a variate if DATA is a one-dimensional data structure like a variate or text, or in a pointer containing a variate for each dimension if DATA is a multi-dimensional data structure like a matrix or table
CLASSIFICATION = *pointers*	Saves the classifying factors for a DATA table, in the same order as the corresponding variates in the LOCATIONS and LEVELS pointers
LEVELS = *pointers*	Saves the levels of the classifying factors where FIND occurs as one of the values of a DATA table, the information is saved in a pointer containing a variate for each factor

Description

The GETLOCATIONS directive finds where a particular item occurs as one of the values stored by a Genstat data structure. So, for example, it can locate the lines within a text structure that are equal to a particular string, or it can locate the rows and columns of a matrix that hold a particular number, or it can locate the numbers of the suffixes where a pointer contains a particular identifier.

The item to find is specified by the FIND parameter, as a scalar (for a number), or a single-valued text (for a string of characters), or a single-valued pointer (for the identifier of a data structure). The data structure to search is supplied by the DATA parameter.

If the FIND pointer contains a dummy, GETLOCATIONS usually looks to see that dummy is contained in the DATA pointer. Alternatively, if you set option SUBSTITUTE=yes and the FIND pointer contains a dummy, it is replaced by the data structure to which it points. Then if that data structure is a dummy, it too is replaced, amd so on until we reach a data structure that is *not* a dummy. However, if the original dummy (or any of the dummies to which it points) is unset, the original dummy is retained. The same substitution is done on any dummies in the DATA pointer. So, when SUBSTITUTE=yes, GETLOCATIONS matches the structures to which the dummies (eventually) point, rather than the dummies themselves.

The number of times that FIND occurs in DATA can be saved, in a scalar, by the NLOCATIONS parameter. The locations where FIND occurs can be saved by the LOCATIONS parameter. These are saved in a variate if DATA is a one-dimensional structure i.e. a scalar, variate, text, factor or pointer. If DATA is a rectangular, diagonal or symmetric matrix, they are saved in a pointer

containing two variates. The first saves the row locations, and the second saves the column locations. If DATA is a table, LOCATIONS saves a pointer with a variate for each factor classifying the table. Each variate stores the locations within the dimension classified by a particular factor. So, for example, with a two-way DATA table the first variate stores the row numbers, and the second variate stores the column numbers. The CLASSIFICATION parameter can save the factors in the same order as the LOCATIONS variates, in case you are unsure of which factor corresponds to each dimension. The LEVELS parameter provides an alternative to LOCATIONS for tables, storing the factor levels corresponding to the positions in each dimension, rather than the numbers of e.g. the rows or columns. If a DATA table has margins and the number to FIND occurs in one of them, a missing value will be stored for the corresponding location or level.

Options: CASE, TOLERANCE, SUBSTITUTE.
Parameters: DATA, FIND, NLOCATIONS, LOCATIONS, CLASSIFICAION, LEVELS.

Action with RESTRICT
GETLOCATIONS takes account of restrictions on DATA variates, factors or texts.

See also
Directives: TXFIND, TXPOSITION.
Functions: GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

GETATTRIBUTE

Accesses attributes of structures.

Option

ATTRIBUTE = *string tokens*	Which attributes to access (nvalues, nlevels, nrows, ncolumns, type {type number}, levels, labels {of a factor or pointer}, nmv, present, identifier, refnumber {structure number}, extra, decimals, characters, minimum, maximum, restriction, mode {integer code 1 - 5 denoting type of values: double real, real, integer, character and word}, maxline {of a text or factor}, rows, columns, classification, margins {of a table}, associatedidentifier {of a table}, unknown {cell of a table}, suffixes {of a pointer}, owner, terms {of an SSPM}, groups {of an SSPM}, weights {of an SSPM}, SSPMauxiliary, SSPrst, tsmmodel, rstat {of an RSAVE}, stype {type as a character string}, referencelevel {of a factor}, drepresentation, unitlabels {of a vector}, iprint, datavariate {of a table}, summarytype {of a table}, percentquantile {of a table of quantiles}, %margin {of a table of percentages}, coding {of a text}); default * i.e. none

Parameters

STRUCTURE = *identifiers*	Structures whose attributes are to be accessed
SAVE = *pointers*	Pointer to store copies of the attributes of each structure; these are labelled by the ATTRIBUTE strings

Description

The GETATTRIBUTE directive allows you to access attributes of each of the structures that are listed with its STRUCTURE parameter. It refers to the list of structures by pointers, which are set up by the SAVE parameter. You must always set the option and both parameters.

If you request an attribute that is not relevant to a structure, it is omitted from the pointer. Thus for example the nlevels, levels and labels settings are relevant only for factors, and nrows and ncolumns only for matrices. The references to the relevant attributes that you specify are always stored in the order shown in the definition of the ATTRIBUTE option above.

For attributes that are single numbers, the information is copied into an unnamed scalar which is pointed to by the appropriate element of the pointer; if the attribute has not been set, then the corresponding scalar will contains a missing value. For the attributes stype, identifier, iprint, margins, associatedidentifier, summarytype and tsmmodel, the corresponding element of the pointer is a text structure containing a single line. For the other attributes, the corresponding element of the pointer stores a reference to the attribute itself. One example is the labels vector of a factor. However, if the factor has no labels vector the corresponding entry of the pointer is set to the missing value.

The type setting gives a scalar value indicating the type of structure, by the code:

1 scalar
2 factor
3 text
4 variate
5 matrix

6 diagonal matrix
7 symmetric matrix
8 table
9 ASAVE
10 TSAVE
11 expression
12 formula
13 dummy
14 pointer
15 LRV
16 SSPM
17 TSM
18 RSAVE
22 tree
32 VSAVE

Alternatively, the `stype` setting obtains the type name in a text structure. This works not only for the standard Genstat types, such as variates and factors, but also for user-defined types (see `STRUCTURE`).

Option: `ATTRIBUTE`.
Parameters: `STRUCTURE`, `SAVE`.

See also
Directives: `SCALAR`, `VARIATE`, `TEXT`, `FACTOR`, `MATRIX`, `SYMMETRICMATRIX`, `DIAGONALMATRIX`, `TABLE`, `DUMMY`, `POINTER`, `EXPRESSION`, `FORMULA`, `LRV`, `SSPM`, `TREE`, `TSM`.
Genstat Reference Manual 1 Summary section on: Data structures.

GRAPH

Produces scatter and line graphs on the terminal or line printer (synonym of LPGRAPH).

Options

CHANNEL = *scalar*	Channel number of output file; default is current output file
TITLE = *text*	General title; default *
YTITLE = *text*	Title for y-axis; default *
XTITLE = *text*	Title for x-axis; default *
YLOWER = *scalar*	Lower bound for y-axis; default *
YUPPER = *scalar*	Upper bound for y-axis; default *
XLOWER = *scalar*	Lower bound for x-axis; default *
XUPPER = *scalar*	Upper bound for x-axis; default *
MULTIPLE = *variate*	Numbers of plots per frame; default * i.e. all plots are on a single frame
JOIN = *string token*	Order in which to join points (ascending, given); default asce
EQUAL = *string tokens*	Whether/how to make bounds equal (no, scale, lower, upper); default no
NROWS = *scalar*	Number of rows in the frame; default * i.e. determined automatically
NCOLUMNS = *scalar*	Number of columns in the frame; default * i.e. determined automatically
YINTEGER = *string token*	Whether y-labels integral (yes, no); default no
XINTEGER = *string token*	Whether x-labels integral (yes, no); default no

Parameters

Y = *identifiers*	Y-coordinates
X = *identifiers*	X-coordinates
METHOD = *string tokens*	Type of each graph (point, line, curve, text); if unspecified, poin is assumed
SYMBOLS = *factors* or *texts*	For factor SYMBOLS, the labels (if defined), or else the levels, define plotting symbols for each unit, whereas a text defines textual information to be placed within the frame for METHOD=text or the symbol to be used for each plot for other METHOD settings; if unspecified, * is used for points, with integers 1-9 to indicate coincident points, ' and . are used for lines and curves
DESCRIPTION = *texts*	Annotation for key

Description

The GRAPH directive has been replaced by the LPGRAPH directive, and may be removed in a future release or modified to produce high-resolution plots instead of character-based plots.

The simplest form of the GRAPH directive produces a point plot (or scatterplot as it is sometimes called). It can also be used to plot lines and curves, and text can be added for extra annotation. The data are supplied as y- and x-coordinates in separate parameter lists. For example

```
VARIATE [VALUES=-16,-7,9,16,7,-8,-12,-5,0,10,4,-4,-3,3,16] X
& [VALUES=0,-14,-12.5,0,14,0,12,0,-10,-9,5,6,-6,-1.5,16] Y
GRAPH Y; X
```

Here the identifiers Y and X are variates of equal length; Genstat uses their values in pairs to give

the coordinates of the points to be plotted.

By default, if you specify several identifiers, Genstat plots them all in the same frame a pair at a time; for example

```
GRAPH Y[1...3]; X[1,2]
```

superimposes plots of `Y[1]` against `X[1]`, `Y[2]` against `X[2]`, and `Y[3]` against `X[1]`. The usual rules governing the parallel expansion of lists apply here: the length of the `Y` parameter list determines the number of plots within the frame, and the `X` parameter list is recycled if it is shorter. To generate several frames from one `GRAPH` statement you can use the `MULTIPLE` option, described below.

The identifiers supplied by the `Y` and `X` parameters need not be variates, but can be any numerical structures: scalars, variates, factors, tables or matrices. The only constraints are that the pairs of structures must have the same numbers of values, and that tables must not have margins.

There are four types of graph available, controlled by the `METHOD` parameter: `point` (the default), `line`, `curve` and `text`.

A `line` plot is one in which each point is joined to the next by a straight line. Alternatively, using the `curve` method, cubic splines are used to produce a smoothed curve through the data points. This does not represent any model fitted in the statistical sense, but as long as the data points are not too widely spaced (especially where the gradient changes quickly) the plotted curve should be a good representation of the underlying function.

By default, Genstat sorts the data so that the x-values are in ascending order before any line or curve is drawn through the points. However, if you set option `JOIN=given`, the points are joined in the order in which they occur in the data; if there are then any missing values there will be breaks in the line at each missing unit.

Plots produced with `METHOD` set to either `line` or `curve` do not include markings for the data points themselves; you should plot these separately if they are required. For example

```
VARIATE [VALUES=-0.1,0.1...0.9] V
& [VALUES=5.5,9.9,8.7,2.3,1.3,5.5] W
GRAPH W,W; V; METHOD=curve,point
```

Here `W` is plotted against `V` twice, first with the `curve` method and then with the `point` method. It is best to plot the line first, so that the symbols for individual points will overwrite those used for the line or curve.

The fourth plotting method is `text`. You can use this to place an item of text within a graph as extra annotation. For example:

```
SCALAR Xt,Yt; VALUE=20,10
TEXT [VALUES='Y=aX+b'] T
GRAPH Y,Yt; X,Xt; METHOD=line,text; SYMBOLS=*,T
```

This plots a line, defined by the variates `Y` and `X`, as described above. In addition, the text `T` is printed within the frame starting at the coordinates defined by the scalars `Yt` and `Xt`. As these statements show, the `SYMBOLS` parameter then specifies the text that is to be plotted. The text is truncated as necessary, if positioned too close to the edge of the graph.

With other methods `SYMBOL` defines the plotting symbol to be used to mark either points or lines on the graph. The default symbol for points is the asterisk, and for lines is a combination of dots and single quotes. If several points coincide, Genstat replaces the asterisk by a digit between 2 and 9, representing the number of coincidences, with 9 meaning nine or more. For point plots, the `SYMBOLS` parameter can be set to either a text or a factor. If you specify a text with a single string, the string is used to label every point; otherwise, the text must have one string for each point.

Normally, output goes to the current output channel, but you can use the `CHANNEL` option to direct it to another. For example, when you are working interactively, you might want to send a graph to a secondary output file so that you can print it later. Unlike some directives you

cannot save the output in a text structure.

The TITLE option lets you set an overall title for the graph. For example:

```
GRAPH [XTITLE='Nitrogen Applied (kg/ha)'] Yield; Nitrogen
```

You can also have individual axis titles, specified by the YTITLE and XTITLE options. Genstat prints the y-axis title as a column of characters down the left-hand side of the graph. New lines are ignored, so that strings within a text are concatenated. Genstat truncates the title if necessary: the maximum possible number of characters is the number of rows of the frame plus 4. The x-axis title is printed below the graph; the maximum number of characters is the number of columns of the frame plus four: long strings are truncated whereas short strings are centred.

If no titles are set, a simple key will be produced below the graph which lists the identifiers and plotting symbols for each pair of Y and X structures. You can obtain your own key by setting the DESCRIPTION parameter, which supplies a line of text for each plot.

By default, Genstat automatically calculates the extent of the axes from the data to be plotted, in such a way that all the data are contained within the frame. You can set one or more of the bounds for the axes by options YLOWER, YUPPER, XLOWER and XUPPER. By setting the upper bound of an axis to a value that is less than the lower bound, you can reverse the usual convention for plotting in which the y-values increase upwards and the x-values increase to the right. Setting the options YINTEGER and XINTEGER constrains the axis markings to be integral, if possible.

The EQUAL option allows you to place constraints on the bounds for the axes. The default setting no (meaning no constraint) uses the boundary values as set by the options or calculated from the data. The settings lower and upper constrain the lower or upper bounds of the two axes to be equal: for example, to plot the line $y=x$ along with the data, setting EQUAL=lower will ensure that it will pass through the bottom left-hand corner of the frame. The scale setting adjusts the y-bounds and x-bounds so that the physical distance on one axis corresponds as closely as possible to physical distance on the other: for example, so that one centimetre will represent the same distance along each axis.

Normally each GRAPH statement produces one frame, and Genstat sets the size so that it will fill one screen or line-printer page, based on the settings of WIDTH and PAGE from OPEN or OUTPUT, or their defaults if these have not been specified. When output is to a file the graph will be placed on a new page, unless this has been disabled using OUTPUT, JOB or SET. The size of the graph is defined in terms of the number of characters in each row and the number of rows in the frame, a row being one line of output. You can adjust the size of the frame by using the NROWS and NCOLUMNS options; the minimum allowed is three rows and three columns, and the maximum number of columns is 17 characters less than the width of the output channel (to leave room for axis markings and titles). There is no maximum on the number of rows. By default, the number of columns is 101, subject to the maximum above, and the number of rows is the number of lines per page, less 8, to allow room for annotation. By defining the page size in advance you can avoid having to specify the numbers of rows and columns when you wish to plot many graphs.

The automatic axis scaling aims to find axis markings that are at reasonable values, but because the markings appear at fixed character positions this may not always be possible. If both upper and lower axis bounds are set, or EQUAL is set in conjunction with axis bounds, or you have requested integral axis markings, there may be conflicting constraints on the axis scaling. If the resultant axis markings then require several decimal places, you may be able to obtain better values by slight adjustments to the numbers of rows or columns.

The MULTIPLE option lets you generate several frames (separate graphs) from one statement. If there is room, the graphs can be printed alongside each other, for example to produce a two-by-two array of plots on a line-printer page. The option should be set to a variate whose elements define the number of graphs to plot in each frame and the number of values in the variate determines the number of frames to be output. For example,

```
GRAPH [MULTIPLE=!(2,1,2)] A,B,C,D,E; X[1...3]
```

will produce three frames; the first containing A against X[1] and B against X[2], the second containing C against X[3] and the third containing D against X[1] and E against X[2]. The sum of the values in the MULTIPLE list gives the total number of structures required to form the plots, which must therefore be equal to the length of the Y parameter list. The X list will be recycled if necessary, as here.

By default, each graph will fit the page (as if it had been produced by an individual GRAPH statement). However, if you set the NCOLUMNS option to a suitably small value, Genstat may be able to fit more than one frame across the page. The MULTIPLE option will then produce the graphs side by side. Remember that 17 columns are automatically added to provide annotation, and five blank columns are used to separate multiple graphs in parallel. This means that, for example, setting NCOLUMNS=20 will produce two graphs in parallel on a screen of width 80, and three graphs when output to a file of width 121 or more.

Options: CHANNEL, TITLE, YTITLE, XTITLE, YLOWER, YUPPER, XLOWER, XUPPER, MULTIPLE, JOIN, EQUAL, NROWS, NCOLUMNS, YINTEGER, XINTEGER.
Parameters: Y, X, METHOD, SYMBOLS, DESCRIPTION.

Action with RESTRICT

You can arrange to plot only a subset of the points specified by a particular pair of Y and X vectors (i.e. variates and/or factors), by restricting either one of them. If both are restricted, then they must be restricted in exactly the same way.

See also

Directives: DGRAPH, D3GRAPH, LPGRAPH.
Genstat Reference Manual 1 Summary section on: Graphics.

GROUPS

Forms a factor (or grouping variable) from a variate or text, together with the set of distinct values that occur.

Options

PRINT = *string token*	Printed output required (summary); default * i.e. no printing
NGROUPS = *scalar*	Number of groups to form when LIMITS is not specified; if NGROUPS is also unspecified, each distinct value (allowing for rounding) defines a group; default *
LMETHOD = *string token*	Defines how to form the levels variate if the setting of the VECTOR parameter is a variate, or the labels if it is a text; if LMETHOD=* no levels/labels are formed, and existing levels (for a variate VECTOR) or labels (for a text VECTOR) of an already declared FACTOR will be retained if still appropriate (given, minimum, median, maximum, limit); default medi
DECIMALS = *scalar*	Number of decimal places to which to round the VECTOR before forming the groups; default * i.e. no rounding
BOUNDARIES = *string token*	Whether to interpret the LIMITS as upper or lower boundaries (upper, lower); default lowe
REDEFINE = *string token*	Whether to allow a structure in the FACTOR list that has already been declared (e.g. as a variate or text) to be redefined (yes, no); default no
CASE = *string token*	Whether the case of letters (small and capital) in text should be regarded as significant or ignored (significant, ignored); default sign
LDIRECTION = *string token*	How to define the levels (for a variate VECTOR) or labels (for a text VECTOR) when LMETHOD = minimum, median or maximum (ascending, given); default asce
OMITUNBOUNDED = *string token*	Whether to omit the (unbounded) group that occurs below the lowest limit when BOUNDARIES=lower, or above the final limit when BOUNDARIES=upper (yes, no); default no

Parameters

VECTOR = *variates* or *texts*	Vectors whose values are to define the groups
FACTOR = *factors*	Structures to be defined as factors to save details of the groups; default * will, if REDEFINE=yes, cause the corresponding VECTOR itself to be defined as a factor
LIMITS = *variates* or *texts*	Limits to define the groups
LEVELS = *variates*	Variate to define the levels of each FACTOR if LMETHOD=give, or to save them otherwise
LABELS = *texts*	Text to define the labels of each FACTOR if LMETHOD=give, or to save them otherwise

Description

The GROUPS directive is designed to form factors from variates or texts. The variates and texts are specified by the VECTOR parameter, and the factors by the FACTOR parameter. With the simplest use of GROUPS you need specify no more than that, and each factor is defined to have a level for every distinct value of its corresponding variate or text. You need not have declared

the factor already; it will be declared automatically if necessary.

Alternatively, you can divide the values of the variate or text into groups to be represented by the factor. You can use the LIMITS parameter to specify the range of values for each group. The limits vector is a text or a variate, depending whether the factor is being defined from a variate or a text; its values specify boundaries for the ranges. The BOUNDARIES option controls whether these are regarded as upper or lower boundaries; by default BOUNDARIES=lower. You can also ask GROUPS itself to set limits that will partition the units into groups of nearly equal size. You should then specify the NGROUPS option and leave the LIMITS parameter unset. (If you give both LIMITS and NGROUPS, then NGROUPS is ignored.)

If you are defining a factor from a variate VECTOR, the LMETHOD option controls how the levels vector is formed, with the following settings:

median	forms the levels from the median of the units in each group (default);
minimum	forms them from the minimum value in each group;
maximum	form them from the maximum value;
limit	uses the values in the LIMITS variate;
given	uses the values supplied (in a variate) by the LEVELS parameter.

With any of the settings median, minumum, maximum or limit, you can use the LEVELS parameter to specify a variate to store the levels that are produced; this can be done even if no factor is being formed, that is if no identifier is supplied for the factor by the FACTOR list. Finally, if you set LMETHOD=*, no levels are formed and any existing levels of the factor will be retained if they are still appropriate; otherwise the levels will be the integers 1 upwards. With any of these settings, you can use the LABELS parameter to specify labels for the factor.

Similar rules apply if you have a text VECTOR except that LMETHOD then governs how the labels are defined for the factor, and LEVELS can be used to specify its levels. The CASE option controls whether the case of the letters in the text strings is important. So, for example, if you set CASE=ignored the strings 'April' and 'april' will be put into the same group. With the default, CASE=significant, they would form different groups.

When the levels are formed from a LIMITS variate, there will be one group with no corresponding limit. If BOUNDARIES=upper, the extra group is above the final limit. The level assigned to that group is then the value that is the same distance above the final limit as the distance between the final limit and the last but one limit. If BOUNDARIES=lower, the extra group is below the first limit, and its level is given the value that is the same distance below the first limit as the distance between the first and second limits. The situation is similar with a LIMITS text, but the label for the extra group is always the single-character string '-'. If you would prefer to have an exact correspondence between the level and the limits, you can set option OMITUNBOUNDED=yes to omit the "unbounded" extra group. Any units beyond the final upper limit, or below the initial lower limit, are then given missing values.

The LDIRECTION option controls the ordering of the levels (for a variate VECTOR) or the labels (for a text VECTOR) when LMETHOD is set to median, minimum or maximum. By default, they are sorted into ascending order, but you can set LDIRECTION=given to take them in the order in which they occur in the VECTOR. This may be useful, for example, if a text vector contains the names of days or of months in calendar order.

You can set the DECIMALS option to request that the values of a variate VECTOR be rounded to a particular number of decimal places before the groups are formed: for example DECIMALS=0 would round each value to the nearest integer.

You can redefine a VECTOR structure as a factor by setting option REDEFINE=yes and omitting to specify any corresponding identifier in the FACTOR list. This can be very useful on occasions when you are unable to define in advance which levels will occur in a set of data.

The PRINT option can be set to summary to print a summary of the contents of the FACTOR

(numbers of values, missing values and levels).

Options: PRINT, NGROUPS, LMETHOD, DECIMALS, BOUNDARIES, REDEFINE, CASE, LDIRECTION, OMITUNBOUNDED.
Parameters: VECTOR, FACTOR, LIMITS, LEVELS, LABELS.

Action with RESTRICT
GROUPS takes account of any restrictions on variates or texts in the VECTOR list, and will give missing values to the excluded units. If more than one vector is restricted, then each of their restrictions must be the same.

See also
Directives: FACTOR, VARIATE, TEXT.
Procedures: FACAMEND, FACDIVIDE, FACPRODUCT, FACSORT, FACLEVSTANDARDIZE, FACUNIQUE, FMFACTORS, FFREERESPONSEFACTOR, QFACTOR.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

HCLUSTER

Performs hierarchical cluster analysis.

Options

PRINT = *string tokens*	Printed output required (dendrogram, amalgamations); default * i.e. no printing
METHOD = *string token*	Criterion for forming clusters (singlelink, nearestneighbour, completelink, furthestneighbour, averagelink, mediansort, groupaverage); default sing
CTHRESHOLD = *scalar*	Clustering threshold at which to print formation of clusters; default * i.e. determined automatically

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix for each cluster analysis
GTHRESHOLD = *scalars*	Grouping threshold where groups are formed from the dendrogram
GROUPS = *factors*	Stores the groups formed
PERMUTATION = *variates*	Permutation order of the units on the dendrogram
AMALGAMATIONS = *matrices*	To store linked list of amalgamations

Description

The aim of cluster analysis is to arrange the *n* sampling units into more or less homogeneous groups. HCLUSTER offers several possibilities. The general strategy is best appreciated in geometrical terms, with the *n* sampling units represented by points in a multidimensional space. In *agglomerative* methods, these points initially represent *n* separate clusters, each containing one member. At each of $n-1$ stages, two clusters are fused into one bigger cluster, until at the final stage all units are fused into a single cluster: this process can be represented by a hierarchical tree whose nodes indicate what fusions have occurred. The methods fuse the two closest clusters and vary in how *closest* is defined. In *single-linkage* cluster analysis, *closest* is defined as the smallest distance between any two samples from different clusters; in *centroid* clustering it is the smallest distance between cluster centroids; and so on. Genstat can display the tree fitted to a given similarity matrix, and provides a scale to show the level of similarity at which the fusions have occurred; such a scaled tree is termed a *dendrogram*.

The input for HCLUSTER is provided by the SIMILARITY parameter, as a list of symmetric matrices, one for each analysis. These matrices can be formed by FSIMILARITY, by HREDUCE or by CALCULATE. Missing values are allowed in the similarity matrix only with the single-linkage method.

A hierarchical tree does not by itself provide a classification. This can be derived by cutting the dendrogram at some arbitrary level of similarity, specified as a percentage similarity using the GTHRESHOLD parameter. Each cluster then consists of those samples occurring on the same detached branch of the dendrogram, and the resulting cluster membership can be saved in a factor whose identifier is specified by the GROUPS parameter. The factor will be declared implicitly, if necessary, and it will have its number of levels set to the number of clusters formed and its number of values taken from the number of rows of the corresponding symmetric matrix. GTHRESHOLD and GROUPS must be either both present or both absent.

The endpoints of the dendrogram correspond to the units in some permuted order. The PERMUTATION parameter allows you to specify a variate to save this order, for example to use in the FSIMILARITY directive. Genstat will define it to be a variate automatically, if necessary, with number of values is taken from the number of rows of the corresponding similarity matrix. Conventionally, the first unit on the dendrogram is unit 1 and so the first value of the variate of

permutations will be 1.

The AMALGAMATIONS parameter can specify a matrix to store information about the order in which the units form groups, and at what level of similarity. At any stage in the process of agglomeration, each group is represented by the unit with the smallest unit number: for example, a group containing units 2, 5, 17 and 22 is represented by unit 2. This means that the final merge is always between a group indexed by unit 1 and a group indexed by another unit. Since there are $n-1$ stages of agglomeration, the matrix will have a number of rows one less than the number of rows of the input similarity matrix. Each row represents a joining of two groups and consists of three values. The first two values are the numbers indexing the two groups that are joining, and the third value is the level of similarity. So the matrix has three columns. The matrix will be declared implicitly, if necessary.

HCLUSTER can print two pieces of information. The first gives details of each amalgamation, followed by a list of clusters that are formed at decreasing levels of similarity. The second is the dendrogram. The PRINT option allows you to control which of these are printed. If METHOD=singlelink and the PRINT setting includes amalgamations, the minimum spanning tree will be printed instead of the stages at which the clusters merge. This is because information from forming the minimum spanning tree is used to form the single linkage clustering.

Alternatively, if you save the AMALGAMATIONS matrix, you can use procedure DDENDROGRAM to display the dendrogram using high-resolution graphics. Also the HFCLUSTERS procedure can be used to obtain the full set of clusters constructed during the cluster analysis, and the similarity values at which they were formed.

The METHOD option has seven possible settings; these determine how the similarities amongst clusters are redefined after each merge. The default singlelink, which has synonym nearestneighbour, gives single linkage. The setting completelink (synonym furthestneighbour) defines the distance between two clusters as the maximum distance between any two units in those clusters. The setting averagelink defines the similarity between a cluster and two merged clusters as the average of the similarities of the cluster with each of the two. For groupaverage, an average is taken over all the units in the two merged clusters. Median sorting is best thought of in terms of clusters being represented by points in a multidimensional space; when two clusters join, the new cluster is represented by the midpoint of the original cluster points.

The CTHRESHOLD option is a scalar which allows you to define the levels of decreasing similarity at which the lists of clusters are printed with their membership. The decreasing levels of similarity are formed by repeatedly subtracting the CTHRESHOLD value from the maximum similarity of 100%. For example, setting CTHRESHOLD=10 will list the clusters formed at 90% similarity, 80%, and so on. At each level, those units that have not joined any group are also listed. If you do not set this option, the default value will be calculated from the range of similarities at which merges occur, to give between 10 and 20 separate levels.

Options: PRINT, METHOD, CTHRESHOLD.
Parameters: SIMILARITY, GTHRESHOLD, GROUPS, PERMUTATION, AMALGAMATIONS.

See also

Directives: FSIMILARITY, HDISPLAY, HLIST, HSUMMARIZE, CLUSTER, HREDUCE.
Procedures: DDENDROGRAM, DCLUSTERLABELS, DMST, BCLASSIFICATION, BKEY, CINTERACTION, HBOOTSTRAP, HCOMPAREGROUPINGS, HFAMALGAMATIONS, HFCLUSTERS, HPCLUSTERS, MASCLUSTER.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

HDISPLAY

Displays results ancillary to hierarchical cluster analyses: matrix of mean similarities between and within groups, a set of nearest neighbours for each unit, a minimum spanning tree, and the most typical elements from each group.

Option

PRINT = *string tokens*	Printed output required (neighbours, tree, typicalelements, gsimilarities); default tree

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix for each cluster analysis
NNEIGHBOURS = *scalars*	Number of nearest neighbours to be printed
NEIGHBOURS = *matrices*	Matrix to store nearest neighbours of each unit
GROUPS = *factors*	Indicates the groupings of the units (for calculating typical elements and mean similarities between groups)
TREE = *matrices*	To store the minimum spanning tree (as a series of links and corresponding lengths)
GSIMILARITY = *symmetric matrices*	To store similarities between groups

Description

You can use the HDISPLAY directive to print ancillary information useful for interpreting cluster analyses, and to save information to use elsewhere in Genstat, for example for plotting.

The SIMILARITIES parameter specifies a list of symmetric similarity matrices. These are operated on, in turn, to produce the output requested by the PRINT option and to save the information specified by other parameters. Since the interpretations of the remaining parameters are closely linked to the different settings of the PRINT option, each setting is discussed below with the relevant parameters.

The NNEIGHBOURS parameter gives a list of scalars indicating how many neighbours will appear in the printed table of nearest neighbours.

The NEIGHBOURS parameter can specify a list of identifiers to store details of nearest neighbours. These will be declared implicitly, if necessary, as matrices. The rows of the matrices correspond to the units; there should be an even number of columns. The values in the odd-numbered columns represent the neighbouring units in order of their similarity, while the values in the even-numbered columns are the corresponding similarities. If you have declared the matrix previously and it does not have enough columns, then NEIGHBOURS stores as many neighbours as possible. If there is an odd number of columns in the matrix, the last column is not filled. If the matrix is declared implicitly, the number of columns will be twice the value of the NNEIGHBOURS scalar.

If the PRINT option includes the setting neighbours, Genstat prints a table of nearest neighbours for every sample, together with their values of similarity. The number of neighbours printed is determined by the value of the NNEIGHBOURS scalar; if NNEIGHBOURS is not set, the table is not printed. This information is also useful for interpreting clusters and ordinations.

The GROUPS parameter specifies a factor to divide the units of each similarity matrix into clusters. You may have formed the factor from a previous hierarchical cluster analysis, using HCLUSTER. This parameter must be set if the PRINT option includes the settings typicalelement or gsimilarities.

If the PRINT option includes the setting typicalelement, Genstat prints the average similarity of each group member with the other group members. This is to help you identify typical members of each group: typical members will have relatively large average similarities compared to those of the other members. Within each group, members are printed in decreasing

order of average similarity.

The GSIMILARITY parameter specifies a list of symmetric matrices in which you can save the mean between-group and within-group similarities. Any structure that you have not declared already will be declared implicitly to be a symmetric matrix with number of rows equal to the number of levels of the factor in the GROUPS parameter.

If the PRINT option includes the setting gsimilarities, Genstat prints the mean similarities between-groups and within-groups. Self-similarities are excluded.

The TREE parameter can specify a matrix to save the minimum spanning tree. The matrix is set up with two columns and number of rows equal to the number of units. For each unit, the value in the first column is the unit to which that unit is linked on its left; the second column is the corresponding similarity. The first unit is not linked to any unit on its left, as it is always the first unit on the tree; so the first row of the matrix contains missing values. The HFAMALGAMATIONS procedure can use the tree to form an amalgamations matrix, representing how the clusters would be formed with this similarity matrix by single-linkage cluster analysis.

Setting the PRINT option to tree prints the minimum spanning tree associated with the similarity matrix specified the SIMILARITY parameter. The minimum spanning tree (MST) is not a Genstat structure, but it can be kept in the form described above: that is, in a matrix with two columns. An MST is a tree connecting the *n* points of a multidimensional representation of the sampling units. In a tree every unit is linked to a connected network and there are no closed loops; the special feature of the MST is that, of all trees with a sampling unit at every node, it is the one whose links have minimum total length. The links include all those that join nearest neighbours; the MST is closely related to single linkage hierarchical trees. Minimum spanning trees are also useful if you superimpose them on ordinations to reveal regions in which distance is badly distorted (see procedure DMST); if neighbouring points, as given by the MST, are distant in the ordination then something is badly wrong.

Option: PRINT.
Parameters: SIMILARITY, NNEIGHBOURS, NEIGHBOURS, GROUPS, TREE, GSIMILARITY.

See also
Directives: HCLUSTER, HLIST, HSUMMARIZE.
Procedures: DDENDROGRAM, DMST, HPCLUSTERS.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

HELP

Provides help information about Genstat commands and functions.

No options

Parameter

TOPIC = *texts*	Single-valued texts indicating the command or function about which the information is required

Description

HELP has a parameter, called TOPIC, which you use to list the commands and functions about which you want information. In Genstat *for Windows*, HELP opens the Windows on-line help file at the appropriate pages.

Options: none.
Parameter: TOPIC.

See also

Procedures: LIBHELP, LIBEXAMPLE.

HISTOGRAM

Produces histograms of data on the terminal or line printer (synonym of LPHISTOGRAM).

Options

CHANNEL = *scalar*	Channel number of output file; default is the current output file
TITLE = *text*	General title; default *
LIMITS = *variate*	Variate of group limits for classifying variates into groups; default *
NGROUPS = *scalar*	When LIMITS is not specified, this defines the number of groups into which a data variate is to be classified; default is the integer value nearest to the square root of the number of values in the variate
LABELS = *text*	Group labels
SCALE = *scalar*	Number of units represented by each character; default 1

Parameters

DATA = *identifiers*	Data for the histograms; these can be either a factor indicating the group to which each unit belongs, a variate whose values are to be grouped, or a one-way table giving the number of units in each group
NOBSERVATIONS = *tables*	One-way table to save numbers in the groups
GROUPS = *factors*	Factor to save groups defined from a variate
SYMBOLS = *texts*	Characters to be used to represent the bars of each histogram
DESCRIPTION = *texts*	Annotation for key

Description

The HISTOGRAM directive has been replaced by the LPHISTOGRAM directive, and may be removed in a future release or modified to produce high-resolution plots instead of character-based plots.

Histograms provide quick and simple visual summaries of data values. The data are divided into several groups, which are then displayed as a histogram consisting of a line of asterisks for each group. The number of asterisks in each line is proportional to the number of values assigned to that group; this figure is also printed at the beginning of each line. The data for the histogram are specified using the DATA parameter in either variates, factors or one-way tables.

If a histogram is to be formed from a variate, Genstat sorts its values into groups as defined by upper and lower bounds. You can also specify a list of variates, to obtain a parallel histogram. For each group one row of asterisks is printed for each variate, labelled by the corresponding identifier. The variates are sorted according to the same intervals; there is no need for them all to have the same numbers of values.

With variates of data, you can use the NGROUPS option to specify the number of groups in the histogram; Genstat will then work out appropriate limits, based on the range of the data, to form intervals of equal width. For example:

```
HISTOGRAM [NGROUPS=5] Data
```

Alternatively, you can define the groups explicitly, by setting the LIMITS option to a variate containing the group limits. For example:

```
VARIATE [VALUES=1,2,3,5,7,8,10] Glimits
HISTOGRAM [LIMITS=Glimits] Data
```

Glimits is a variate with seven values, producing a histogram in which the data are split into eight groups; ≤1, 1-2, 2-3, 3-5, 5-7, 7-8, 8-10, >10. The upper limit of each group is included

within that group, so the group 3-5, for example, contains values that are greater than 3 and less than or equal to 5. The values of the limits variate are sorted into ascending order if necessary, but the variate itself is not changed.

You can use the LABELS option to provide your own labelling for the groups of the histogram. It should be set to a text vector of length equal to the number of groups. If neither NGROUPS nor LIMITS has been set, the number of groups is determined from the number of values in the LABELS structure. If LABELS is also unset, the default number of groups is chosen as the integer value nearest to the square root of the number of values, up to a maximum of 10. Alternatively, procedure AKAIKEHISTOGRAM provides a more sophisticated method of generating histograms, using Akaike's Information Criterion (AIC) to generate an optimal grouping of the data.

The data for the histogram can also be specified as a factor (which defines the assignment of each unit to a group of the histogram). Genstat then counts the number of units that occur with each level of the factor; thus the number of groups of the histogram is the number of levels of the factor and the value for each group is the corresponding total. If the LABELS option is unset, the labels of the factor (if present) are used to label the groups, otherwise Genstat uses the factor levels.

When Genstat plots the histogram of a one-way table, the number of groups is the number of levels of the factor classifying the table and the values of the table indicate the number of observations in each group. If the LABELS option is unset, the labels or levels of the classifying factor are again used to label the histogram.

When producing a parallel histogram the data structures must all be of the same type: variate, factor or table. If parallel histograms are to be formed from several factors, they must all have the same number of levels, and the labels or levels of the first factor will be used to identify the groups. Likewise, if you are forming parallel histograms from several tables, they must all have the same number of values, and the classifying factor of the first table will define the labelling of the histogram.

The SYMBOLS parameter can specify alternative plotting characters to be used instead of the asterisk. For example:

```
HISTOGRAM Variate; SYMBOLS='+'
```

You can specify a different string for each structure in a parallel histogram. If you specify strings of more than one character, Genstat uses the characters in order, recycled as necessary, until each histogram bar is of the correct length.

The TITLE option lets you set an overall title for the output, and the DESCRIPTION parameter can be used to provide a text for labelling the histogram instead of the identifiers of the DATA structures.

Normally one asterisk will represent one unit. However, if there are many data values and the groups become large, Genstat may not be able to fit enough asterisks into one row. It will then alter the scaling so that one asterisk represents several units. You can set the scaling explicitly using the SCALE option; the value specified is rounded to the nearest integer, and determines how many units should be represented by each asterisk.

HISTOGRAM has two output parameters that allow you to save information that has been generated during formation of the histogram. The NOBSERVATIONS parameter allows you to save a one-way table of counts that contains the number of observations that were assigned to each group; the missing-value cell of this table will contain a count of the number of units that were missing and that therefore remain unclassified. When producing a histogram from a variate, you can use the GROUPS parameter to specify a factor to record the group to which each unit was allocated.

Normally, output goes to the current output channel, but you can use the CHANNEL option to direct it to another. For example, when you are working interactively, you might want to send a graph to a secondary output file so that you can print it later. Unlike some directives (for example, PRINT) you cannot save the output in a text structure.

Options: CHANNEL, TITLE, LIMITS, NGROUPS, LABELS, SCALE.
Parameters: DATA, NOBSERVATIONS, GROUPS, SYMBOLS, DESCRIPTION.

Action with RESTRICT
You can restrict a DATA variate or factor to form a histogram for only a subset of the units. However, the restriction does not carry over to any other variates or factors listed by the DATA parameter.

See also
Directives: BARCHART, DHISTOGRAM, LPHISTOGRAM.
Genstat Reference Manual 1 Summary section on: Graphics.

HLIST

Lists the data matrix in abbreviated form.

Options

GROUPS = *factor*	Defines groupings of the units; used to split the printed table at appropriate places and to label the groups; default *
UNITS = *text* or *variate*	Names for the rows (i.e. units) of the table; default *

Parameters

DATA = *variates* or *factors*	The data variables
TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is taken

Description

HLIST lists the values of the data matrix in a condensed form, either in their original order or, more usefully, in the order determined by a cluster analysis (see HCLUSTER). This representation can be very helpful for revealing patterns in the data, associated with clusters, or for an initial scan of the data to pick out interesting features of the variables.

The DATA parameter specifies a list of variates or factors, all of which must be of the same length. The TEST parameter specifies a list of strings, one for each variate or factor in the DATA parameter list, to define the "type" of each one. This is similar to the TEST parameter used in FSIMILARITY to determine how differences in variate or factor values for each unit contribute to the overall similarity between units. However, HLIST distinguishes only between qualitative variables (factors or variates with settings simplematching - rogerstanimoto) and quantitative variables (variates with other settings). The values of qualitative variates are printed directly. If the range of a quantitative variate is greater than 10, the printed values are scaled to lie in the range 0 to 10. This scaling is done by subtracting the minimum value, dividing by the range and then multiplying by 10. If the range is less than 10, the values are printed unscaled; so quantitative variates with values that are all less than 1 will appear as 0 in the abbreviated table. The values are printed with no decimal places, and in a field-width of 3.

The RANGE parameter contains a list of scalars, one for each variable in the DATA list. This allows you to check that the values of each variable lie within the given range. The range is also used to standardize quantitative variates, so that you can impose a standard range for example when variates are measured on commensurate scales. You can omit the RANGE parameter for all or any of the variables by giving a missing identifier or a scalar with a missing value; Genstat then uses the observed range.

The UNITS option allows you to change the labelling of the units in the table; you can specify a text or a pointer or a variate.

You can use the GROUPS option to specify a factor that will split the units into groups. The table from HLIST is then divided into sections corresponding to the groups. If the factor has labels, these are used to annotate the sections; otherwise a group number is used.

Options: GROUPS, UNITS.

Parameters: DATA, TEST, RANGE.

Action with RESTRICT
You can restrict any of the DATA variates or factors to list only a subset of the units. If more than one of these is restricted, then they must all be restricted to the same set of units.

See also
Directives: HCLUSTER, HDISPLAY, HSUMMARIZE.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

HREDUCE

Forms a reduced similarity matrix (referring to the GROUPS instead of the original units).

Options

PRINT = *string token*	Printed output required (similarities); default * i.e. no printing
METHOD = *string token*	Method used to form the reduced similarity matrix (first, last, mean, minimum, maximum, zigzag); default firs

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix
REDUCEDSIMILARITY = *symmetric matrices*	Output (reduced) similarity matrix
GROUPS = *factors*	Factor defining the groups
PERMUTATION = *variates*	Permutation order of units (for METHOD = firs, last or zigz)

Description

Sometimes you may want to regard an n-by-n similarity matrix S as being partitioned into b-by-b rectangular blocks. You might then want to form a reduced matrix of similarities, between the different blocks instead of between the individual units. To do this you have to arrange for each of the b^2 blocks of the full matrix to be replaced by a single value. Each diagonal block must be replaced by unity. The METHOD option specifies how to replace the off-diagonal blocks, for example the maximum, minimum or mean similarity within the block. The *zigzag* method (Rayner 1966) is relevant in particular when the data consist of b soil samples for each of which information is recorded on several soil horizons, possibly different in the different samples. The method recognizes that certain horizons might be absent from some soil samples; this leads to finding successive optimal matches, conditional on the constraint that one horizon cannot match a horizon that has already been assigned to a higher level; after finding these optima, an average is taken for each horizon.

The SIMILARITY parameter specifies the similarity matrix for the full set of n observations; this must be present and have values. The REDUCEDSIMILARITY parameter specifies an identifier for the reduced similarity matrix, of order b; this will be declared implicitly if you have not declared it already. The factor that defines the classification of the units into groups must be specified by the GROUPS parameter. The units can be in any order, so that for example the units of the first group need not be all together nor given first. The labels of the factor label the reduced similarity matrix.

The PERMUTATION parameter, if present, must specify a variate. It defines the ordering of samples within each group, and so must be specified for methods first, last and zigzag. Within each group, the unit with the lowest value of the permutation variate is taken to be the first sample, and so on. Genstat will, if necessary, use a default permutation of one up to the number of rows of the similarity matrix.

If you set option PRINT=similarities, the values of the reduced symmetric matrix are printed, as percentages.

(Note: this directive was originally called REDUCE.)

Options: PRINT, METHOD.
Parameters: SIMILARITY, REDUCEDSIMILARITY, GROUPS, PERMUTATION.

Reference

Rayner, J.H. (1966). Classification of soils by numerical methods. *Journal of Soil Science*, **17**, 79-92.

See also

Directive: `FSIMILARITY`.

Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

HSUMMARIZE

Forms and prints a group by levels table for each test together with appropriate summary statistics for each group.

Option

GROUPS = *factor*	Factor defining the groups; no default i.e. this option must be specified

Parameters

DATA = *variates* or *factors*	The data variables
TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is taken

Description

The HSUMMARIZE directive helps you to see which clusters, if any, are distinguished by each variable. It requires a factor to define the clusters, as well as the original DATA variables (variates or factors), together with their types and, optionally, their ranges. From this it prints a frequency table for each variable, classified by the grouping factor and the different values of the variable concerned.

The option and parameters of the HSUMMARIZE directive are the same as those of the HLIST directive, and are described there.

For qualitative variables (variates or factors with TEST settings simplematching - rogerstanimoto) the values are integral, and for each group Genstat calculates an interaction statistic labelled chi-square. This statistic does not have a significance level attached to it, but it does draw attention to groups for which the distribution is markedly different from the overall distribution.

For quantitative variables (i.e. variates with other settings) values are rounded to the nearest point on an 11-point scale (0-10). The interaction statistic is analogous to Student's t, and it draws attention to the groups for which the mean value is markedly different from the overall mean (again with no significance level attached). Missing values are ignored in the computation of these statistics.

Option: GROUPS.
Parameters: DATA, TEST, RANGE.

Action with RESTRICT

You can restrict any of the DATA variates or factors to do the calculations for only a subset of the units. If more than one of these is restricted, then they must all be restricted to the same set of units.

See also

Directives: HCLUSTER, HDISPLAY, HLIST.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

IF

Introduces a block-if control structure.

No options

Parameter

expression	Logical expression, indicating whether or not to execute the first set of statements.

Description

A *block-if* structure consists of one or more alternative sets of statements. The first of these is introduced by an IF statement. There may then be further sets introduced by ELSIF statements. Then you can have a final set introduced by an ELSE statement, and the whole structure is terminated by an ENDIF statement. Thus the general form is:

first

```
IF expression
  statements
```

then either none, one, or several blocks of statements of the form

```
ELSIF expression
  statements
```

then, if required, a block of the form

```
ELSE
  statements
```

and finally the statement

```
ENDIF
```

Each expression must evaluate to a single number, which is treated as a logical value: a zero value is treated as *false* and non-zero as *true*. Genstat executes the block of statements following the first true expression. If none of the expressions is *true*, the block of statements following ELSE (if present) is executed.

You can thus use these directives to built constructs of increasing complexity. The simplest form would be to have just an IF statement, then some statements to execute, and then an ENDIF. For example:

```
IF MINIMUM(Sales) < 0
  PRINT 'Incorrect value recorded for Sales.'
ENDIF
```

If the variate Sales contains a negative value, the PRINT statement will be executed. Otherwise Genstat goes straight to the statement after ENDIF.

To specify two alternative sets of statements, you can include an ELSE block. For example

```
IF Age < 20
  CALCULATE Pay = Hours*1.75
ELSE
  CALCULATE Pay = Hours*2.5
ENDIF
```

calculates Pay using two different rates: 1.75 for Age less than 20, and 2.5 otherwise.

Finally, to have several alternative sets, you can include further sets introduced by ELSIF statements. Suppose that we want to assign values to X according to the rules:

X=1 if Y=1
X=2 if Y ≠ 1 and Z=1
X=3 if Y ≠ 1 and Z=2

X=4 if Y ≠ 1 and Z ≠ 1 or 2

This can be written in Genstat as follows:

```
IF Y == 1
  CALCULATE X = 1
ELSIF Z == 1
  CALCULATE X = 2
ELSIF Z == 2
  CALCULATE X = 3
ELSE
  CALCULATE X = 4
ENDIF
```

If Y is equal to 1, the first CALCULATE statement is executed to set X to 1. If Y is not equal to 1, Genstat does the tests in the ELSIF statements, in turn, until it finds a *true* condition; if none of the conditions is *true*, the CALCULATE statement after ELSE is executed to set X to 4. Thus, for Y=99 and Z=1, Genstat will find that the condition in the IF statement is *false*. It will then test the condition in the first ELSIF statement; this produces a *true* result, so X is set to 2. Genstat then continues with whatever statement follows the ENDIF statement. Block-if structures can be nested to any depth, to give conditional constructs of even greater flexibility.

Options: none.
Parameter: unnamed.

See also
Directives: ELSIF, ELSE, ENDIF, EXIT, CASE, CALCULATE.
Genstat Reference Manual 1 Summary section on: Program control.

INPUT

Specifies the input file from which to take further statements.

Options

PRINT = *string tokens*	What output to generate from the statements in the file (statements, macros, procedures, unchanged); default stat
REWIND = *string token*	Whether to rewind the file (yes, no); default no

Parameter

scalar	Channel number of input file

Description

Having opened a file of Genstat statements on another input channel (for example by the OPEN directive) you can switch control to that channel at any time using an INPUT statement. You specify the channel as a number or as a scalar containing that number. For example,

```
OPEN 'MYPROCS.GEN'; CHANNEL=4; FILETYPE=input
INPUT 4
```

The file can contain any valid Genstat statements: they will be executed just as if they had been on the original input channel. In this file you could use an INPUT statement to switch back to channel 1 after a while. Alternatively, you may have set up several input files and jump from one to another, again using INPUT. You can use RETURN to go back to the previous channel or STOP to end this run of Genstat. If the end of the file is reached without finding any of these statements, control will be passed back to the previous input channel as explained in the description of the RETURN directive. Note that if you use INPUT to go back to an earlier channel, you may affect the way in which RETURN works (again see the description of RETURN).

The PRINT option can be used to specify whether the statements read from the file should be echoed to the current output channel. This is used in the same way as INPRINT in JOB and SET.

The REWIND option allows you to return to the beginning of the file. You might need to do this, for example, if you had made an error, so that the statements on the secondary input file were executed wrongly. After correcting your error you could set REWIND=yes to start again from the beginning of the file.

Options: PRINT, REWIND.
Parameter: unnamed.

See also

Directives: RETURN, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

INTERPOLATE

Interpolates values at intermediate points.

Options

`CURVE` = *string token*	Type of curve to be fitted to calculate the interpolated value (`linear, cubic`); default `line`
`METHOD` = *string token*	Type of interpolation required (`interval, value, missing`): for `METHOD=valu`, values are interpolated for each point in the `NEWINTERVAL` variate and stored in the `NEWVALUE` variate; for `METHOD=inte`, points are estimated in the `NEWINTERVAL` variate for the observations in the `NEWVALUE` variate; while for `METHOD=miss`, the `NEWVALUE` and `NEWINTERVAL` lists are irrelevant, `INTERPOLATE` now interpolates for missing values in the `OLDVALUE` and `OLDINTERVAL` variates (except those missing in both variates); default `inte`

Parameters

`OLDVALUES` = *variates*	Observations from which interpolation is to be done
`NEWVALUES` = *variates*	Results of each interpolation
`OLDINTERVALS` = *variates*	Points at which each set of `OLDVALUES` was observed
`NEWINTERVALS` = *variates*	Points for each set of `NEWVALUES`

Description

If you have a set of pairs of observations (x, y), you can use interpolation to estimate either a value y for a value x that need not be in the set, or a value x for a value y that likewise need not be in the set. The simplest way to interpolate is by joining successive pairs of observations by straight lines and reading off the appropriate values in between: then the two cases are called *linear interpolation* (obtaining y from x) and *inverse linear interpolation* (obtaining x from y). Genstat can alternatively join the points by cubic functions instead of straight lines. Genstat uses the term *values* to describe the set of y-values and *intervals* for the set of x-values, no matter whether you are doing direct or inverse interpolation.

Genstat does the interpolation for each parallel set of variates in the parameter lists. Each variate in the `OLDINTERVALS` list specifies the x-values of a set of observed points; the corresponding variate in the `OLDVALUES` list specifies the corresponding y-values. The variates in the `NEWINTERVALS` and `NEWVALUES` lists are for the x-values and y-values of the interpolated points.

If you set `METHOD=value`, Genstat does ordinary interpolation, and you use the `NEWINTERVALS` variate to specify the x-values for which you require interpolated y-values. Genstat calculates the y-values and stores them in the corresponding `NEWVALUES` variate; this variate will be declared implicitly if you have not declared it already.

For the interpolation to take place, the x-values must be in either monotonically increasing or decreasing order; thus, if necessary, Genstat takes a copy of the x-values and y-values and sorts these (in parallel) to put the x-values into ascending order.

Assume that wheat plants have been sampled on five occasions and their growth stage (Zadoks) assessed. `INTERPOLATE` interpolates values, which it stores in variate `Nzad`, to estimate the growth stage that the plant has reached after 50, 100 and 150 days.

```
VARIATE [NVALUES=6] Zadoks,Days; \
  VALUES=!(0,15,23,35,65,95),!(0,50,84,119,147,182)
& [NVALUES=3] Nzadoks,Ndays; VALUES=!(25,50,75),!(50,100,150)
INTERPOLATE [METHOD=value] Zadoks; NEWVALUES=Nzad; \
```

```
    OLDINTERVALS=Days; NEWINTERVALS=Ndays
```

Similarly, if you set `METHOD=interval`, Genstat does inverse interpolation. You must then specify the *y*-values in the `NEWVALUES` variate. Genstat calculates the *x*-values and stores them in the corresponding `NEWINTERVALS` variate, which will be declared implicitly if necessary. Again the *x*-values must be in monotonically increasing or decreasing order, and Genstat will produce a sorted copy if necessary. Inverse interpolation is the default.

This statement would use inverse linear interpolation to estimate how long after planting we have to wait for the plant to reach growth stages 25, 50 and 75 Zadoks.

```
INTERPOLATE [METHOD=interval] Zadoks; NEWVALUES=Nzadoks; \
  OLDINTERVALS=Days; NEWINTERVALS=Nd
```

If you set `METHOD=missing`, Genstat ignores the `NEWVALUES` and `NEWINTERVALS` parameters; it estimates values for *x* or *y* when the other is missing, placing the results in the previously missing position of the `OLDVALUES` or the `OLDINTERVALS` variates. Ordinary interpolation is used when the missing value is in *y*, and inverse interpolation when it is in *x*. If both the *x*-value and the *y*-value are missing for a particular unit, no values can be interpolated for it, and it remains missing. To do linear interpolation requires that both the *x*-value and the *y*-value should be non-missing for the point on each side of the unit with the missing value. For cubic interpolation, there must be two non-missing points on each side of the unit.

The `CURVE` option has two settings, `linear` and `cubic`. By default, `CURVE=linear`, and successive pairs of observations are connected by straight-line segments for linear, or inverse-linear, interpolation. For cubic interpolation you set `CURVE=cubic`; there must then be at least four values in each of the `OLDVALUES` and `OLDINTERVALS` variates.

For linear & inverse linear interpolation between variates you can use the `VINTERPOLATE` procedure.

Options: `CURVE`, `METHOD`.
Parameters: `OLDVALUES`, `NEWVALUES`, `OLDINTERVALS`, `NEWINTERVALS`.

Action with `RESTRICT`

Either or both of the `OLDVALUES` and `OLDINTERVALS` variates can be restricted to arrange for only a subset of the observed points to be used. Similarly, either or both of the `NEWVALUES` and `NEWINTERVALS` variates can be restricted to arrange that values are calculated for only a subset of units of the new variates.

See also

Directives: `KRIGE`, `COKRIGE`.
Procedure: `VINTERPOLATE`.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

IRREDUNDANT

Forms irredundant test sets for the efficient identification of a set of objects.

Options

PRINT = *string tokens*	Controls printed output (numbers, diagram, notdistinguished, messages); default numb, diag, notd, mess
BESTSET = *pointer*	Saves the best set
SETS = *matrix*	Saves details of the available sets
NOTDISTINGUISHED = *matrix*	Saves details of the objects that cannot be distinguished
METHOD = *string token*	Algorithm to use (exact, sequential); default exac
TAXONNAMES = *text*, *variate* or *factor*	Defines labels for the objects (or *taxa*) to be identified; default uses the unit labels vector of the CHARACTER factors
GROUPS = *factor*	Defines groupings of the objects so that the sets are constructed to distinguish only between the objects that belong to different groups; default constructs sets to distinguish between individual objects
OBJECT = *scalar* or *text*	If this is specified, sets are constructed just to distinguish the specified object (or *taxon*) from the other objects
NDISTINCTIONS = *scalar*	Number of factors required in each set to distinguish between each pair of objects; default 1
MAXPREFERENCE = *scalar*	Maximum preference of the factors to be included in the sets
MAXSIZE = *scalar*	Limit on number of factors in a set (sets containing more than this are discarded); default * i.e. none
NPRINT = *scalar*	Number of sets to print (a positive number specifies the number to print, a negative number sets a tolerance on the difference between the sizes of the sets printed and the size of the best set); default * prints them all
NSAVE = *scalar*	Number of sets to save in the SETS matrix; default * saves them all
LIMSETS = *variate*	Variate containing two numbers n_1 and n_2, if this is specified then every time that there are more than n_1 sets under construction using the exact method, the sets are arranged in order of increasing size and all sets containing more factors than set n_2 are deleted
DISTINCTIONS = *string token*	Whether or not to store the distinctions or recalculate them at every stage in the exact algorithm (store, calculate); default stor
CRITERION = *string token*	Function to be use to select factors by the sequential method (ndistinctions, weightedndistinctions); default ndis
MAXCYCLE = *scalar*	Maximum number of improvement cycles to perform during the sequential method; default 20
EQUIVALENCE = *scalar*	Value for determining equivalence of the selection criteria of tests selected during the sequential method

Parameters

CHARACTER = *identifiers*	Factors, and/or tables classified by a single factor,

	defining the properties of the objects to to be identified
COST = *scalars*	Cost associated with each factor; default 1
PREFERENCE = *scalar*	Preference rating for each factor (1 representing most preferred etc.); default 1
VARIABLE = *scalar* or *text*	Factor level used to represent variable information; default is to use a missing value
INAPPLICABLE = *scalar* or *text*	Factor level used to indicate that the information provided by that factor is inapplicable for a particular object

Description

The IRREDUNDANT directive is useful when you have a set of objects (or *taxa*) whose properties can be described by a set of discrete-valued tests. Many applications are biological. For example, in botanical work, the taxa may be species of plant and the tests may require the observation of characters like the colours of petals or numbers of leaves. Similarly, in microbiology, the tests may involve the ability of an organism to grow in various media. IRREDUNDANT helps you to select an efficient set of tests that can be applied, in a batch, to identify any unknown specimen of any of the objects. (The batch of tests is then often printed as a diagnostic table; see Payne & Preece 1980.) As all the tests in the set are to be used for every identification, it is best for the set to contain as few tests as possible. So there should thus be no *redundant* tests: these are tests that can be deleted from the set without causing any object (or *taxon*) to be no longer identifiable. Sets of tests that contain no redundant tests are known as *irredundant*.

Consider taxa A, B, C and D, whose responses to tests 1-5 are shown in the table below. The symbol "+", for example in the entry for taxon A and test 1, indicates that all specimens of taxon A will always give a positive result to test 1, the symbol "–" for taxon D with test 1 indicates a negative result, and the symbol "v" for taxon B with test 3 indicates that some specimens of D will give a positive result to test 3 but others will give a negative result.

	Test				
Taxon	1	2	3	4	5
A	+	+	+	–	+
B	+	–	v	–	–
C	+	–	–	+	+
D	–	+	+	+	–

The table contains several irredundant sets, one of which contains the tests 1, 3 and 5. (If, for example, test 3 is deleted from this set, taxa A and C can no longer be distinguished). Another set contains tests 2 and 4. So, the irredundant sets can be of different sizes. The optimum set will often be defined to be one containing a minimum number of tests. Alternatively, if the test cost different amounts to apply, the optimum set may be one with minimum total cost. However, whichever of these situations applies, the optimum set will be irredundant, as otherwise a better set could be obtained by deleting a redundant test.

The characteristics of the taxa and tests are specified using the CHARACTER parameter of IRREDUNDANT. In the simplest situation, this provides a list of factors, one for each test (or character), as with the BKEY procedure. The factors contain a unit for each taxon, and the level stored in that unit indicates how the taxon can respond to the test. For example, irredundant sets for the data in the table above could be constructed as follows:

```
TEXT [VALUES=A,B,C,D] Taxa
FACTOR [NVALUES=4; LEVELS=2] T1,T2,T3,T4,T5
READ T1,T2,T3,T4,T5
2  2  2  1  2
2  1  *  1  1
```

```
2  1  1  2  2
1  2  2  2  1  :
IRREDUNDANT [TAXONNAMES=Taxa] T1,T2,T3,T4,T5
```

Level 1 of the factors `T1` - `T5` represents a negative response, and level 2 represents a positive response. The variable response of taxon B with test 3 is represented by a missing value, but you can use the `VARIABLE` parameter to use a particular level of the factor instead. There may be tests that are not applicable to some of the taxa. For example, when identifying insects, tests concerning colours of wings are not applicable to those that do not fly! The level to be used to indicate these responses is specified by the `INAPPLICABLE` parameter. Costs for the test can be specified by the `COST` parameter; by default, these are all taken to be one. Names for the taxa can be supplied, in either a text or a variate or a factor, using the `TAXONNAMES` parameter. If this is not set, `IRREDUNDANT` uses the unit labels of the `CHARACTER` factors if any have been defined (see the `FACTOR` directive), or otherwise the integers 1, 2 upwards.

The use of the `VARIABLE` option works well with responses that are completely variable i.e. where the specimens of the taxon may give any of the available results to the test. However, when the tests have more than two possible results, there may be taxa that can give some but not all of the available results to a test. The responses to a test like this should be specified by a two-way table classified by one factor with a level for each possible result, and another with a level for each taxon. The table should then contain a positive (e.g. one) whererever the taxon concerned can deliver the result, and zero elsewhere. For example suppose that, with test `T6`, taxon A, C and D always give result 1, 2 and 3 respectively, but taxon B can give either or results 2 or 3. The relevant table could then be constructed and used as follows:

```
FACTOR [LABELS=Taxa] Taxfact
FACTOR [LEVELS=3] T6fact
TABLE  [CLASSIFICATION=T6fact,Taxfact; VALUES=\
       " level 1:"  1, 0, 0, 0, \
       " level 2:"  0, 1, 1, 0, \
       " level 3:"  0, 1, 0, 1 ] T6tab
IRREDUNDANT [TAXONNAMES=Taxfact] T1,T2,T3,T4,T5,T6tab
```

The standard irredundant sets contain at least one test to distinguish each pair of taxa. However, to guard against mistakes in either the original data on during the subsequent use of the set, you can set the `NDISTINCTIONS` option to ask for the set to include a larger number of tests able to distinguish each pair. Another refinement is that you can set the `GROUPS` option to a factor defining groupings of the taxa. The sets are then formed to distinguish only pairs of taxa that belong to different groups. Alternatively, you may want a set of tests to either confirm whether or not the specimen belongs to one particular taxon. The taxon of interest should then be indicated by setting the `OBJECT` option to the number of the taxon or, if textual taxon names have been defined, to the text identifying the taxon. Finally, if you set both `GROUPS` and `OBJECT`, the sets will be constructed to confirm whether or not a specimen belongs to a particular group.

`IRREDUNDANT` provides two methods for constructing the irredundant sets. The default is to use an exact method (Payne 1991) which constructs all possible sets for the dataset concerned. However, with some datasets, there may be too many sets to construct them all. If you run out of workspace (or time), you can use the `LIMSETS` to specify a variate containing two integers n_1 and n_2. Then whenever there are more than n_1 sets under construction, the sets are arranged in order of increasing size and all sets containing more factors than set n_2 are deleted. The method then no longer guarantees to find all the irredundant sets containing the fewest number of tests or with the minimum total costs, but in the situations where this modification is needed, it is very unlikely that it will fail to find any of them.

Alternatively, you can set option `METHOD=sequential` to use a sequential algorithm (Payne & Preece 1980, Section 6.6). This does not guarantee to find a set with minimum size or cost, but it takes much less computing time and should always should produce a satisfactory set. The sequential method starts with an initial set containing all the essential tests, and then adds

additional tests, one at a time, until each pair of taxa can be distinguished. (A test is *essential* if it is the only test which can distinguish between a particular pair of taxa.) The criterion for selecting the test to add to the set at each stage is usually the number of pairs of taxa that the test distinguishes, of those pairs not distinguished by tests already in the set. If costs have been defined, this number of pairs is divided by the cost of the test concerned.

Setting option `CRITERION=weighted` uses a refinement, suggested by Barnett *et al*. (1983), which weights each pair of taxa by the reciprocal of the number of tests that can distinguish between them. The criterion is then the maximum *weighted* number of pairs of taxa (divided by the cost of the test, if defined). This causes tests that distinguish "difficult" pairs of taxa (those with nearly identical characteristics) to be selected earlier during the construction of the set, and thus tends to generate smaller sets. You can set a preference rating for each test using the `PREFERENCE` parameter; the most-preferred tests should have ratings of one, and less-preferred tests should have ratings of two and upwards. Then, if at any stage there is then more than one test with the best criterion value, the most-preferred test is selected. If these preferances are especially important, you may also also want to set the `EQUIVALENCE` option to a scalar, *e* say. Then all tests whose criterion values are within *e* of the current maximum are regarded as equivalent, and the best test is selected from within these tests according to the preferences.

The main disadvantage of most sequential methods is that they produce only a single set of tests. In order to allow a choice of sets and as a way of improving the original set, `IRREDUNDANT` can run through a sequence of cycles. In each of these, the tests in the best set are deleted in turn, further tests are selected to separate the pairs of taxa distinguished only by the deleted test, and any redundant tests are deleted. If no improvement is achieved, all the non-essential tests are deleted, and the set is reformed without using those tests. The process can be then repeated until no improvements are being achieved of until the number of cycles exceeds the setting of the `MAXCYCLE` option (default 20).

Printed output is controlled by the `PRINT` option, with settings:

`numbers`	numbers of the tests in the sets,
`diagram`	table showing the contents of the sets,
`notdistinguished`	lists of pairs of taxa that cannot be distinguished,
`messages`	messages for example when the number of sets has been reduced as requested by the `LIMSETS` option, or concerning pairs of taxa than cannot be distinguished.

The default is `PRINT=numb,diag,notd,mess`.

The best set can be saved using the `BESTSET` option, as a pointer containing the relevant factors. The `SETS` option can save a matrix, with a row for each set and a column for each test, representing all the sets that have been formed. In each row the matrix generally stores the number one in the columns corresponding to the tests in that set, and zero elsewhere. However, if the sets have been constructed to confirm the identification of a single taxon, the matrix contains more informative numbers than one. So, down each column wherever one would be stored, it instead stores the level given by the taxon for the factor corresponding to the test concerned. The `NOTDISTINGUISHED` option can save information about the pairs of taxa that cannot be distinguished, or that are distinguished by less than `NDISTINCTIONS` tests. The matrix has a row for each such pair of taxa, and three columns. Columns 1 and 2 contain the numbers of the taxa in the pair, and column 3 contains the number of tests that can distinguish them.

Options: `PRINT`, `BESTSET`, `SETS`, `NOTDISTINGUISHED`, `METHOD`, `UNITS`, `GROUPS`, `OBJECT`, `NDISTINCTIONS`, `MAXPREFERENCE`, `MAXSIZE`, `NPRINT`, `NSAVE`, `LIMSETS`, `DISTINCTIONS`, `CRITERION`, `MAXCYCLE`, `EQUIVALENCE`.

Parameters: `CHARACTER`, `COST`, `PREFERENCE`, `VARIABLE`, `INAPPLICABLE`.

Method

The exact method uses an extension of the algorithm of Payne (1991). The sequential method is an extension by Barnett *et al*. (1983) of the algorithm described in Payne & Preece (1980) Section 6.6.

Action with RESTRICT

IRREDUNDANT takes account of restrictions on any of the CHARACTER factors or on TAXONNAMES or GROUPS.

References

Barnett, J.A., Payne, R.W. & Yarrow, D. (1983). *Yeasts: Characteristics and Identification*. Cambridge: Cambridge University Press.

Payne, R.W. (1991). Algorithm AS263 Construction of irredundant test sets. *Applied Statistics*, **40**, 213-229.

Payne, R.W. & Preece, D.A. (1980). Identification keys and diagnostic tables: a review (with discussion). *Journal of the Royal Statistical Society, Series A*, **143**, 253-292.

See also

Procedures: BKEY, IDENTIFY.

Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Multivariate and cluster analysis.

JOB

Starts a Genstat job.

Options

`INPRINT` = *string tokens*	Printing of input as in `PRINT` option of `INPUT` (`statements, macros, procedures, unchanged`); default `unch`
`OUTPRINT` = *string tokens*	Additions to output as in `PRINT` option of `OUTPUT` (`dots, page, unchanged`); default `unch`
`DIAGNOSTIC` = *string tokens*	Defines the least serious class of Genstat diagnostic which should still be generated (`messages, warnings, faults, extra, unchanged`); default `unch`
`ERRORS` = *scalar*	Limit on number of error diagnostics that may occur before the job is abandoned; default `*` i.e. no change
`PROMPT` = *text*	Characters to be printed for the input prompt
`WORDLENGTH` = *string token*	Length of word (8 or 32 characters) to check in identifiers, directives, options, parameters and procedures (`long`, `short`); default `*` i.e. no change

Parameter

text	Name to identify the job

Description

The `JOB` and `ENDJOB` directives can be used to partition a Genstat program into separate *jobs*. A job is a self-contained subsection of a program. All data structures and procedures are lost at the end of each job. Any setting defined by a `UNITS` statement is deleted, as are the special structures set up by analyses like regression and analysis of variance. The graphics environment is also reset to the initial default. Thus, in many ways, it is as though Genstat was starting again for each new job. However, any files that have been attached to Genstat retain their current status from job to job. So, for example, Genstat will continue to add output to the end of an output file, or will continue reading from the current point of an input file.

The `JOB` directive is used to start a new job. It has a parameter which can be set to a text to identify the job (for example in the message at the end of the job), and options to control some aspects of the Genstat "environment". However, Genstat will automatically start a job at the beginning of a program, or after an `ENDJOB` statement, so you do not need to give a `JOB` statement unless you wish to define an identifying text or to modify the environment.

`JOB` also has options that allow you to modify some aspects of the Genstat environment. The default settings of the options will leave these aspects unchanged so, if any aspect is modified, it will remain in that form (unless modified again) in any subsequent job. All these aspects have initial defaults, described below, that apply at the outset of a program. However, it is possible to arrange for Genstat to run commands from a *start-up* file before it executes the first statement of a program, so the initial environment can differ from machine to machine.

The `INPRINT` option specifies which pieces of input from the current input channel will be recorded in the current output file. (The current input channel may be a file or, in an interactive run, it may also be the keyboard.) The settings correspond to three types of input:

`statements`	statements that are typed explicitly on the keyboard or which occur explicitly in an input file,
`macros`	statements or parts of statements that have been supplied in macros, using the ## notation (1.9.2), and
`procedures`	statements occurring within procedures.

The initial default is to record only `statements` for input from a file, or to record nothing if input is from the keyboard. The recording of input can be modified also by the `INPRINT` option of the `SET` directive, or by the `PRINT` option of `INPUT`.

The `OUTPRINT` option controls the way in which the output from many Genstat directives will start: `page` ensures that output to a file will start at the head of a page, and `dots` produces a line of dots beginning with the line number of the statement that has generated the analysis. The initial default is to give a new page and a line of dots if output is to a file, but neither if output is to the screen. This can be modified also by the `OUTPRINT` option of the `SET` directive, or by the `PRINT` option of `OUTPUT`.

The `DIAGNOSTICS` option controls the reporting of errors and possible mistakes. In order of increasing seriousness there three classes of diagnostic: messages, warnings and faults. Messages are comments that are made to draw your attention to things that might need closer investigation, like large residuals in an analysis of variance or a regression. Warnings are definite errors, but ones that are not sufficiently serious to prevent Genstat from continuing; an example would be an attempt to print a data structure with no values. Faults are the most serious type of error. A fault in a batch run will cause Genstat to stop executing the current job. However, Genstat will continue to read and interpret the statements so that it can find the start of the next job (if any); at the same time it will report any further errors that it finds, up to the number specified by the `ERRORS` option.

The setting of `DIAGNOSTICS` indicates the level of stringency to be adopted. Thus, if `DIAGNOSTICS=warnings`, Genstat will report faults and warnings (but not messages), while `DIAGNOSTICS=messages` ensures that all three classes are reported. The setting `extra` is similar to `messages` but will also generate a dump of system information after any fault. You can prevent the output of any diagnostics by putting `DIAGNOSTICS=*`. The initial default is to set `DIAGNOSTICS=messages`.

The `WORDLENGTH` parameter controls the number of characters that are stored and checked in identifiers and names of directives, procedures, options, parameters and functions. In releases prior to 4.2 this was always eight, but from 4.2 onwards you can choose between eight (`WORDLENGTH=short`) and 32 (`WORDLENGTH=long`). The initial default is `long`.

Options: `INPRINT`, `OUTPRINT`, `DIAGNOSTIC`, `ERRORS`, `PROMPT`, `WORDLENGTH`.
Parameter: unnamed.

See also
Directives: `ENDJOB`, `STOP`.
Genstat Reference Manual 1 Summary section on: Program control.

KRIGE

Calculates kriged estimates using a model fitted to the sample variogram.

Options

PRINT = *string token*	Controls printed output (description, search, weights, monitor, data); default desc
Y = *variate*	Y positions (not needed for 2-dimensional regular data i.e. when DATA is a matrix)
X = *variate*	X positions (needed only for 2-dimensional irregular data)
YOUTER = *variate*	Variate containing 2 values to define the Y-bounds of the region to be examined (bottom then top); by default the whole region is used
XOUTER = *variate*	Variate containing 2 values to define the X-bounds of the region to be examined (left then right); by default the whole region is used
YINNER = *variate*	Variate containing 2 values to define the Y-bounds of the interpolated region (bottom then top); no default
XINNER = *variate*	Variate containing 2 values to define the X-bounds of the interpolated region (left then right); no default
BLOCK = *variate*	Dimensions (length and height) of block; default !(0, 0) i.e. punctual kriging
RADIUS = *scalar*	Maximum distance between target point in block and usable data
SEARCH = *string token*	Type of search (isotropic, anisotropic); default isot
MINPOINTS = *scalar*	Minimum number of data points from which to compute elements; default 7
MAXPOINTS = *scalar*	Maximum number of data points from which to compute elements (2 < MINPOINTS ≤ MAXPOINTS < 41); default 20
NSTEP = *scalar*	Number of steps for numerical integration; (3 < NSTEP < 11); default 8
DRIFT = *string token*	Amount of drift (constant, linear, quadratic); default cons
YXRATIO = *scalar*	Ratio of Y interval to X interval; default 1.0
INTERVAL = *scalar*	Distance between successive interpolations; default 1.0

Parameters

DATA = *variates* or *matrices*	Observed measurements as a variate or, for data on a regular grid, as a matrix
ISOTROPY = *string tokens*	Form of variogram (isotropic, Burgess, geometrical); default isot
MODELTYPE = *string tokens*	Model fitted to the variogram (power, boundedlinear, circular, spherical, doublespherical, pentaspherical, exponential, besselk1, gaussian, cubic, stable, cardinalsine, matern); default powe
NUGGET = *scalars*	The nugget variance
SILLVARIANCES = *variates*	Sill variances of the spatially dependent component; default none

RANGES = *variates*	Ranges of the spatially dependent component; default none
GRADIENT = *variates*	Slope of the unbounded component; default none
EXPONENT = *variates*	Power of the unbounded component or power for the stable model; default none
SMOOTHNESS = *scalar*	Value of ν parameter for the Matern model; defalt none
PHI = *variates*	Phi parameters of an anistropic model (ISOTROPY = Burg or geom)
RMAX = *variates*	Maximum gradient or distance parameter of an anistropic model
RMIN = *variates*	Minimum gradient or distance parameter of an anistropic model
PREDICTIONS = *matrices*	Kriged estimates
VARIANCES = *matrices*	Estimation variances
LAGRANGEMULTIPLIER = *matrices* or *pointers*	Saves the Lagrange multipliers from each kriging solution
MEASUREMENTERROR = *scalar*	Specifies the variance of the measurement error
SAVE = *pointers*	Supplies the model name and estimates, as saved from MVARIOGRAM

Description

The KRIGE directive computes the ordinary kriging estimates of a variable at positions on a grid from data and a model variogram. The data must be supplied, using the DATA parameter, in one of the two forms as for the FVARIOGRAM procedure: i.e. for data on a regular grid, in a matrix defined with a variate of column labels to provide the x-values and a variate of row labels to provide the y-values or, for irregularly scattered data, as a variate with the X and Y options set to variates to supply the spatial coordinates.

By default all data are considered when forming the kriging system. However, a subset of the data may be selected by limiting the area to a rectangle defined by XOUTER and YOUTER options. Each of these should be set to a variate with two values to define lower and upper limits in the x (East-West) and y (North-South) directions respectively.

The positions at which *Z* is predicted (estimated) are contained in a rectangle defined by the XINNER, YINNER and INTERVAL options. XINNER and YINNER are set to variates similarly to XOUTER and YOUTER, and their limits should not lie outside those of XOUTER and YOUTER. INTERVAL is set to a scalar to define the distance between the successive positions in the rows and columns of the grid at which kriging is to be done, specified in the same units as the data. However, if the aim is to make a map, INTERVAL should be chosen so that it represents no more than 2 mm on the final printed document. The optimality of the kriging will then not be degraded noticeably by the subsequent contouring.

Kriging may be either punctual, i.e. at "points" which have the same size and shape as the sample support, or on bigger rectangular blocks. The size of the blocks is specified by the BLOCK option, in a variate whose two values define the length of the block first in the x direction (eastings) and then in the y direction (northings). By default the BLOCK variate contains two zero values, to give punctual kriging. The average semivariances between point and block are computed by integrating the variogram numerically over the block. The number of steps in each direction is defined by the NSTEP option. The default of 8 is recommended as a compromise between speed and accuracy. The kriging may be accelerated at the expense of accuracy by reducing NSTEP, or accuracy gained by increasing it. The minimum is 4 and the maximum 10.

The minimum and maximum number of points for the kriging system are set by the MINPOINTS and MAXPOINTS options. There is a minimum limit of 3 for MINPOINTS and a

maximum of 40 for MAXPOINTS, and MINPOINTS must be less than or equal to MAXPOINTS. The defaults are 7 and 20 respectively. Data points may be selected around the point or block to be kriged by setting the RADIUS option to the radius within which they must lie. If the variogram is anisotropic, the search may be requested to be anisotropic by setting option SEARCH to anisotropic; by default SEARCH=isotropic.

Universal kriging may be invoked by setting the DRIFT option to linear or to quadratic, i.e. to be of order 1 or 2 respectively. By default is DRIFT=constant, to give ordinary kriging. For data in a regular grid that is not square, the ratio of the spacing in the y direction to that in the x direction is given by the YXRATIO option. The default is 1.0 for square.

The variogram is specified by its type and parameters. The model and estimates can be saved using the SAVE parameter of MVARIOGRAM, and passed on to KRIGE using its SAVE parameter. Alternatively, they can be supplied as follows.

The model can be defined by setting the MODELTYPE option to either power, boundedlinear (one dimension only), circular, spherical, doublespherical, pentaspherical, exponential, besselk1 (Whittle's function), gaussian, cubic, stable (i.e. powered exponential; see Webster & Oliver 2001), cardinalsine (Chiles & Delfiner 1999) or matern. All models may have a nugget variance, supplied using the NUGGET option; this is the constant estimated by MVARIOGRAM. For punctual kriging, you can specify the variance of any measurement error using the MEASUREMENTERROR parameter. The parameters of the power function (the only unbounded model) are defined by the GRADIENT and EXPONENT parameters. The parameter for the power of the stable model is supplied using the EXPONENT parameter. The parameter ν for the matern function is supplied using the SMOOTHNESS parameter. The simple bounded models, i.e. all other settings of MODELTYPE except doublespherical, require the SILLVARIANCES (the sill of the correlated variance) and RANGES parameters. The latter is strictly the correlation range of the boundedlinear, circular, spherical and pentaspherical models, while for the asymptotic models it is the distance parameter of the model. The doublespherical model requires SILLVARIANCES and RANGES to be set to variates of length two, to correspond to the two components of the model.

The ISOTROPY parameter allows the variation to be defined to be either isotropic or anisotropic in one of two ways: either Burgess anisotropy (Burgess & Webster 1980) or geometric anisotropy (Journel & Huijbregts 1978, Webster & Oliver 1990). The anisotropy is specified by three parameters, namely PHI, the angle in radians of the direction of maximum variation, RMAX, the maximum gradient or distance parameter of the model, and RMIN, the minimum gradient or distance parameter. The power, stable, exponential, Gaussian, pentashperical, spherical, cubic and circular functions may be anisotropic.

KRIGE calculates two matrices, one of predictions (or estimates), which can be saved using the PREDICTIONS parameter, and the other of the prediction (estimation or kriging) variances saved using the VARIANCES parameter. The matrices are arranged with the first row of each matrix at the bottom following geographic rather than mathematical convention. You can save the Lagrange multipliers from the kriging solution using the LAGRANGEMULTIPLIER parameter. For ordinary Kriging the Lagrange multipliers are saved in a matrix (with a multiplier for each point). For universal Kriging a pointer of matrices is saved, where a matrix to save the Lagrange multipliers of each equation term.

The PRINT option can be set to data to print the data (2-dimensional regular data only). It also allows intermediate results to be printed. The setting search lists the results of the search for data around each position to be kriged, weights lists the kriging weights at each position and monitor monitors the formation and inversion of the kriging matrices for each position. These options enable you to check that the kriging is working reasonably. However, they can produce a great deal of output, and should not be requested when kriging large matrices, such as might be wanted for mapping.

Options: PRINT, Y, X, YOUTER, XOUTER, YINNER, XINNER, BLOCK, RADIUS, SEARCH, MINPOINTS, MAXPOINTS, NSTEP, DRIFT, YXRATIO, INTERVAL.

Parameters: DATA, ISOTROPY, MODELTYPE, NUGGET, SILLVARIANCES, RANGES, GRADIENT, EXPONENT, SMOOTHNESS, PHI, RMAX, RMIN, PREDICTIONS, VARIANCES, LAGRANGEMULTIPLIER, MEASUREMENTERROR, SAVE.

Action with RESTRICT

You can restrict any of the DATA variate to do the estimation using only a subset of the units. If more than one of the variates is restricted, they must all be restricted in the same way.

References

Burgess, T.M. & Webster, R. (1980). Optimal interpolation and isarithmic mapping of soil properties. I. The semi-variogram and punctual kriging. *Journal of Soil Science*, **31**, 315-331.

Chiles, J.P. & Delfiner, P. (1999). *Geostatistics: Modeling Spatial Uncertainty*. Wiley, New York.

Journel, A.G. & Huijbregts, C.J. (1978). *Mining Geostatistics*. Academic Press, London.

Webster, R. & Oliver, M.A. (1990). *Statistical Methods in Soil and Land Resource Survey*. Oxford University Press, Oxford.

Webster, R. & Oliver, M.A. (2001). *Geostatistics for Environmental Scientists*. Wiley, Chichester.

See also

Directives: FVARIOGRAM, FCOVARIOGRAM, MCOVARIOGRAM, COKRIGE.

Procedures: MVARIOGRAM, DVARIOGRAM, DCOVARIOGRAM, DHSCATTERGRAM, KCROSSVALIDATION.

Genstat Reference Manual 1 Summary section on: Spatial statistics.

LIST

Lists details of the data structures currently available within Genstat.

Options

PRINT = *string tokens*	What to print (identifier, attributes); default iden, attr
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
SYSTEM = *string token*	Whether to include "system" structures with prefix _ (yes, no); default no
SCOPE = *string token*	When used within a procedure, this allows the listing of structures in the program that called the procedure (SCOPE=external), or in the main program itself (SCOPE=global), rather than those within the procedure (local, external, global); default loca
NSTRUCTURES = *scalar*	Saves the number of structures of the requested types
SAVE = *pointer*	Saves a pointer containing the structures of the requested types

Parameter

strings	Types of structure to list (all, ASAVE, diagonal, dummy, expression, factor, formula, lrv, matrix, pointer, RSAVE, scalar, sspm, symmetric, table, text, tree, TSAVE, tsm, variate, VSAVE); default all

Description

The LIST directive can be used to list the data structures that are currently available. It is particularly useful when you are working interactively to remind you about the data structures that you have set up, and the identifiers that you have used.

The parameter specifies the types of structure that you want to list. By default, all types are listed.

By default LIST prints details of relevant attributes, as well as the identifiers, but this can be controlled using the PRINT option.

The CHANNEL option can be set to a scalar to divert the output to another output channel. Alternatively, it can specify the identifier of text data structure to store the output (and, if you specify an undeclared structure, it will automatically be defined as a text).

The SYSTEM option of LIST controls whether structures whose identifiers begin with the underscore character _ are listed; this character is used as a prefix for the specialized structures set up by the Client program in Genstat *for Windows*, so their inclusion could be confusing.

The SCOPE option can be used within a procedure to list the data structures in the program that called the procedure (SCOPE=external) or in the outermost part of the program (SCOPE=global).

The SAVE option can save a pointer containing the structures of the requested types. This is not formed if there are none.

The NSTRUCTURES option can save a scalar storing the number of structures of these types. (So you can check whether a SAVE pointer has been formed by checking whether NSTRUCTURES is greater than zero.)

Options: PRINT, CHANNEL, SYSTEM, SCOPE, NSTRUCTURES, SAVE.
Parameter: unnamed.

See also
Directives: DUMP, GETATTRIBUTE.
Genstat Reference Manual 1 Summary section on: Data structures.

LPCONTOUR

Produces contour maps of two-way arrays of numbers using character (i.e. line-printer) graphics.

Options

`CHANNEL` = *scalar*	Channel number of output file; default is current output file
`INTERVAL` = *scalar*	Contour interval for scaling; default `*` i.e. determined automatically
`TITLE` = *text*	General title; default `*`
`YTITLE` = *text*	Title for y-axis; default `*`
`XTITLE` = *text*	Title for x-axis; default `*`
`YLOWER` = *scalar*	Lower bound for y-axis; default 0
`YUPPER` = *scalar*	Upper bound for y-axis; default 1
`XLOWER` = *scalar*	Lower bound for x-axis; default 0
`XUPPER` = *scalar*	Upper bound for x-axis; default 1
`YINTEGER` = *string token*	Whether y-labels integral (`yes`, `no`); default `no`
`XINTEGER` = *string token*	Whether x-labels integral (`yes`, `no`); default `no`
`LOWERCUTOFF` = *scalar*	Lower cut-off for array values; default `*`
`UPPERCUTOFF` = *scalar*	Upper cut-off for array values; default `*`

Parameters

`GRID` = *identifiers*	Pointers (of variates representing the columns of a data matrix), matrices or two-way tables specifying values on a regular grid
`DESCRIPTION` = *texts*	Annotation for key

Description

A contour plot provides a way of displaying three-dimensional data in a two-dimensional plot. `LPCONTOUR` produces these in "line-printer" format, that is, using the characters of ordinary output rather than a high-resolution graphics display. It was added in Release 10 as a synonym of the earlier `CONTOUR` directive, which may be modified to use high-resolution graphics in a future release.

The data values are supplied as a rectangular array of numbers that represent the values of the variable in the third dimension, often referred to as *height* or the *z-axis*. The first two dimensions (x and y) are the rows and columns indexing the array; the complete three-dimensional data set is referred to as a *surface* or *grid*. Contours are lines that are used to join points of equal height, and usually some form of interpolation is used to estimate where these points lie. The resulting contour plot is not necessarily very "realistic" when compared to perspective plots produced by `DSURFACE`, but it has the advantage that the entire surface can easily be examined, without the danger of some parts being obscured by high points or regions.

You might use contour plots for example when you have data sampled at points on a regular grid, such as the concentrations of a trace element or nutrient in the soil. Contours are also very useful when fitting nonlinear models, when they can be used to study two-dimensional slices of the likelihood surface, to help find good initial estimates of the parameters.

`LPCONTOUR` produces output by using cubic interpolation between the grid points to estimate a z-value for each character position in the plot. Each value is reduced to a single digit in the range 0 ... 9, according to the rules described below. To produce the contour plot only the even digits are printed: you can then see the contours as the boundaries between the blank areas and the printed digits.

The `GRID` parameter can be set to a matrix, a two-way table (with the first factor defining the

rows), or a pointer to a set of variates each containing a column of data. We explain the conventions in terms of a matrix as input, but similar rules apply to the other structures. When reading or printing a matrix the origin of the rows and columns (row 1, column 1) appears at the top left-hand corner. However, in forming the contour plot the rows are reversed in order so that the first row of the matrix is placed at the bottom of the contour; thus the origin of the contour is located, according to the usual conventions, at the bottom left-hand corner of the plot. The DCONTOUR directive, which plots contours using high-resolution graphics, also reverses the rows of the grid in the same way.

LPCONTOUR scales the grid values by dividing by the contour interval. The scaled grid values are then converted to single digits by taking the remainder modulo 10 and truncating the fractional part. The INTERVAL option allows you to set the contour interval. For example, if the grid values range from 17 to 72 and the interval is set to 10, contour lines (the boundaries between blank space and printed digits) will occur at grid values of 20, 30, 40, 50, 60 and 70. By default, the interval is determined from the range of the data in order to obtain 10 contours.

The UPPERCUTOFF and LOWERCUTOFF options can be used to define a window for the grid values that will form the contours. All values above or below these are printed as X. Setting either UPPERCUTOFF or LOWERCUTOFF will change the default contour interval, as the range of data values is effectively curtailed.

You can use the TITLE, YTITLE and XTITLE option to annotate the contour plot. If you specify several grids, these will be plotted in separate frames and the text of the TITLE option will appear at the top of each one. You should thus use TITLE only to give a general description of what the contours represent. The DESCRIPTION parameter can be used to add specific descriptions to be printed at the bottom of each individual plot.

The YUPPER and YLOWER options allow you to set upper and lower bounds for the y-axis; thus generating axis labels that reflect the range of values over which the grid was observed or evaluated. Setting YINTEGER=yes will ensure the labels are printed as integers, if possible. The default axis bounds are 0.0 and 1.0. The options XLOWER, XUPPER and XINTEGER similarly control labelling of the x-axis.

Options: CHANNEL, INTERVAL, TITLE, YTITLE, XTITLE, YLOWER, YUPPER, XLOWER, XUPPER, YINTEGER, XINTEGER, LOWERCUTOFF, UPPERCUTOFF.
Parameters: GRID, DESCRIPTION.

Action with RESTRICT

LPCONTOUR takes account of restrictions on any of the variates in a GRID pointer.

See also

Directives: DCONTOUR, LPGRAPH, LPHISTOGRAM.
Genstat Reference Manual 1 Summary section on: Graphics.

LPGRAPH

Produces point and line graphs using character (i.e. line-printer) graphics.

Options

CHANNEL = *scalar*	Channel number of output file; default is current output file
TITLE = *text*	General title; default *
YTITLE = *text*	Title for y-axis; default *
XTITLE = *text*	Title for x-axis; default *
YLOWER = *scalar*	Lower bound for y-axis; default *
YUPPER = *scalar*	Upper bound for y-axis; default *
XLOWER = *scalar*	Lower bound for x-axis; default *
XUPPER = *scalar*	Upper bound for x-axis; default *
MULTIPLE = *variate*	Numbers of plots per frame; default * i.e. all plots are on a single frame
JOIN = *string token*	Order in which to join points (ascending, given); default asce
EQUAL = *string tokens*	Whether/how to make bounds equal (no, scale, lower, upper); default no
NROWS = *scalar*	Number of rows in the frame; default * i.e. determined automatically
NCOLUMNS = *scalar*	Number of columns in the frame; default * i.e. determined automatically
YINTEGER = *string token*	Whether y-labels integral (yes, no); default no
XINTEGER = *string token*	Whether x-labels integral (yes, no); default no

Parameters

Y = *identifiers*	Y-coordinates
X = *identifiers*	X-coordinates
METHOD = *string tokens*	Type of each graph (point, line, curve, text); if unspecified, poin is assumed
SYMBOLS = *factors* or *texts*	For factor SYMBOLS, the labels (if defined), or else the levels, define plotting symbols for each unit, whereas a text defines textual information to be placed within the frame for METHOD=text or the symbol to be used for each plot for other METHOD settings; if unspecified, * is used for points, with integers 1-9 to indicate coincident points, ' and . are used for lines and curves
DESCRIPTION = *texts*	Annotation for key

Description

LPGRAPH plots graphs in "line-printer" format, that is, using the characters of ordinary output rather than a high-resolution graphics display. It was added in Release 10 as a synonym of the earlier GRAPH directive, which may be modified to use high-resolution graphics in a future release.

The simplest form of the LPGRAPH directive produces a point plot (or scatterplot as it is sometimes called). It can also be used to plot lines and curves, and text can be added for extra annotation. The data are supplied as y- and x-coordinates in separate parameter lists. For example

```
VARIATE [VALUES=-16,-7,9,16,7,-8,-12,-5,0,10,4,-4,-3,3,16] X
& [VALUES=0,-14,-12.5,0,14,0,12,0,-10,-9,5,6,-6,-1.5,16] Y
```

```
LPGRAPH Y; X
```

Here the identifiers `Y` and `X` are variates of equal length; Genstat uses their values in pairs to give the coordinates of the points to be plotted.

By default, if you specify several identifiers, Genstat plots them all in the same frame a pair at a time; for example

```
LPGRAPH Y[1...3]; X[1,2]
```

superimposes plots of `Y[1]` against `X[1]`, `Y[2]` against `X[2]`, and `Y[3]` against `X[1]`. The usual rules governing the parallel expansion of lists apply here: the length of the `Y` parameter list determines the number of plots within the frame, and the `X` parameter list is recycled if it is shorter. To generate several frames from one `LPGRAPH` statement you can use the `MULTIPLE` option, described below.

The identifiers supplied by the `Y` and `X` parameters need not be variates, but can be any numerical structures: scalars, variates, factors, tables or matrices. The only constraints are that the pairs of structures must have the same numbers of values, and that tables must not have margins.

There are four types of graph available, controlled by the `METHOD` parameter: `point` (the default), `line`, `curve` and `text`.

A `line` plot is one in which each point is joined to the next by a straight line. Alternatively, using the `curve` method, cubic splines are used to produce a smoothed curve through the data points. This does not represent any model fitted in the statistical sense, but as long as the data points are not too widely spaced (especially where the gradient changes quickly) the plotted curve should be a good representation of the underlying function.

By default, Genstat sorts the data so that the x-values are in ascending order before any line or curve is drawn through the points. However, if you set option `JOIN=given`, the points are joined in the order in which they occur in the data; if there are then any missing values there will be breaks in the line at each missing unit.

Plots produced with `METHOD` set to either `line` or `curve` do not include markings for the data points themselves; you should plot these separately if they are required. For example

```
VARIATE [VALUES=-0.1,0.1...0.9] V
& [VALUES=5.5,9.9,8.7,2.3,1.3,5.5] W
LPGRAPH W,W; V; METHOD=curve,point
```

Here `W` is plotted against `V` twice, first with the `curve` method and then with the `point` method. It is best to plot the line first, so that the symbols for individual points will overwrite those used for the line or curve.

The fourth plotting method is `text`. You can use this to place an item of text within a graph as extra annotation. For example:

```
SCALAR Xt,Yt; VALUE=20,10
TEXT [VALUES='Y=aX+b'] T
LPGRAPH Y,Yt; X,Xt; METHOD=line,text; SYMBOLS=*,T
```

This plots a line, defined by the variates `Y` and `X`, as described above. In addition, the text `T` is printed within the frame starting at the coordinates defined by the scalars `Yt` and `Xt`. As these statements show, the `SYMBOLS` parameter then specifies the text that is to be plotted. The text is truncated as necessary, if positioned too close to the edge of the graph.

With other methods `SYMBOL` defines the plotting symbol to be used to mark either points or lines on the graph. The default symbol for points is the asterisk, and for lines is a combination of dots and single quotes. If several points coincide, Genstat replaces the asterisk by a digit between 2 and 9, representing the number of coincidences, with 9 meaning nine or more. For point plots, the `SYMBOLS` parameter can be set to either a text or a factor. If you specify a text with a single string, the string is used to label every point; otherwise, the text must have one string for each point.

Normally, output goes to the current output channel, but you can use the `CHANNEL` option to

direct it to another. For example, when you are working interactively, you might want to send a graph to a secondary output file so that you can print it later. Unlike some directives you cannot save the output in a text structure.

The TITLE option lets you set an overall title for the graph. For example:

```
LPGRAPH [XTITLE='Nitrogen Applied (kg/ha)'] Yield; Nitrogen
```

You can also have individual axis titles, specified by the YTITLE and XTITLE options. Genstat prints the y-axis title as a column of characters down the left-hand side of the graph. New lines are ignored, so that strings within a text are concatenated. Genstat truncates the title if necessary: the maximum possible number of characters is the number of rows of the frame plus 4. The x-axis title is printed below the graph; the maximum number of characters is the number of columns of the frame plus four: long strings are truncated whereas short strings are centred.

If no titles are set, a simple key will be produced below the graph which lists the identifiers and plotting symbols for each pair of Y and X structures. You can obtain your own key by setting the DESCRIPTION parameter, which supplies a line of text for each plot.

By default, Genstat automatically calculates the extent of the axes from the data to be plotted, in such a way that all the data are contained within the frame. You can set one or more of the bounds for the axes by options YLOWER, YUPPER, XLOWER and XUPPER. By setting the upper bound of an axis to a value that is less than the lower bound, you can reverse the usual convention for plotting in which the y-values increase upwards and the x-values increase to the right. Setting the options YINTEGER and XINTEGER constrains the axis markings to be integral, if possible.

The EQUAL option allows you to place constraints on the bounds for the axes. The default setting no (meaning no constraint) uses the boundary values as set by the options or calculated from the data. The settings lower and upper constrain the lower or upper bounds of the two axes to be equal: for example, to plot the line $y=x$ along with the data, setting EQUAL=lower will ensure that it will pass through the bottom left-hand corner of the frame. The scale setting adjusts the y-bounds and x-bounds so that the physical distance on one axis corresponds as closely as possible to physical distance on the other: for example, so that one centimetre will represent the same distance along each axis.

Normally each LPGRAPH statement produces one frame, and Genstat sets the size so that it will fill one screen or line-printer page, based on the settings of WIDTH and PAGE from OPEN or OUTPUT, or their defaults if these have not been specified. When output is to a file the graph will be placed on a new page, unless this has been disabled using OUTPUT, JOB or SET. The size of the graph is defined in terms of the number of characters in each row and the number of rows in the frame, a row being one line of output. You can adjust the size of the frame by using the NROWS and NCOLUMNS options; the minimum allowed is three rows and three columns, and the maximum number of columns is 17 characters less than the width of the output channel (to leave room for axis markings and titles). There is no maximum on the number of rows. By default, the number of columns is 101, subject to the maximum above, and the number of rows is the number of lines per page, less 8, to allow room for annotation. By defining the page size in advance you can avoid having to specify the numbers of rows and columns when you wish to plot many graphs.

The automatic axis scaling aims to find axis markings that are at reasonable values, but because the markings appear at fixed character positions this may not always be possible. If both upper and lower axis bounds are set, or EQUAL is set in conjunction with axis bounds, or you have requested integral axis markings, there may be conflicting constraints on the axis scaling. If the resultant axis markings then require several decimal places, you may be able to obtain better values by slight adjustments to the numbers of rows or columns.

The MULTIPLE option lets you generate several frames (separate graphs) from one statement. If there is room, the graphs can be printed alongside each other, for example to produce a two-by-two array of plots on a line-printer page. The option should be set to a variate whose elements

define the number of graphs to plot in each frame and the number of values in the variate determines the number of frames to be output. For example,

```
LPGRAPH [MULTIPLE=!(2,1,2)] A,B,C,D,E; X[1...3]
```

will produce three frames; the first containing `A` against `X[1]` and `B` against `X[2]`, the second containing `C` against `X[3]` and the third containing `D` against `X[1]` and `E` against `X[2]`. The sum of the values in the `MULTIPLE` list gives the total number of structures required to form the plots, which must therefore be equal to the length of the `Y` parameter list. The `X` list will be recycled if necessary, as here.

By default, each graph will fit the page (as if it had been produced by an individual `LPGRAPH` statement). However, if you set the `NCOLUMNS` option to a suitably small value, Genstat may be able to fit more than one frame across the page. The `MULTIPLE` option will then produce the graphs side by side. Remember that 17 columns are automatically added to provide annotation, and five blank columns are used to separate multiple graphs in parallel. This means that, for example, setting `NCOLUMNS=20` will produce two graphs in parallel on a screen of width 80, and three graphs when output to a file of width 121 or more.

Options: `CHANNEL`, `TITLE`, `YTITLE`, `XTITLE`, `YLOWER`, `YUPPER`, `XLOWER`, `XUPPER`, `MULTIPLE`, `JOIN`, `EQUAL`, `NROWS`, `NCOLUMNS`, `YINTEGER`, `XINTEGER`.
Parameters: `Y`, `X`, `METHOD`, `SYMBOLS`, `DESCRIPTION`.

Action with `RESTRICT`

You can arrange to plot only a subset of the points specified by a particular pair of `Y` and `X` vectors (i.e. variates and/or factors), by restricting either one of them. If both are restricted, then they must be restricted in exactly the same way.

See also

Directives: `DGRAPH`, `D3GRAPH`, `LPCONTOUR`, `LPHISTOGRAM`.
Genstat Reference Manual 1 Summary section on: Graphics.

LPHISTOGRAM

Produces histograms using character (i.e. line-printer) graphics.

Options

CHANNEL = *scalar*	Channel number of output file; default is the current output file
TITLE = *text*	General title; default *
LIMITS = *variate*	Variate of group limits for classifying variates into groups; default *
NGROUPS = *scalar*	When LIMITS is not specified, this defines the number of groups into which a data variate is to be classified; default is the integer value nearest to the square root of the number of values in the variate
LABELS = *text*	Group labels
SCALE = *scalar*	Number of units represented by each character; default 1

Parameters

DATA = *identifiers*	Data for the histograms; these can be either a factor indicating the group to which each unit belongs, a variate whose values are to be grouped, or a one-way table giving the number of units in each group
NOBSERVATIONS = *tables*	One-way table to save numbers in the groups
GROUPS = *factors*	Factor to save groups defined from a variate
SYMBOLS = *texts*	Characters to be used to represent the bars of each histogram
DESCRIPTION = *texts*	Annotation for key

Description

Histograms provide quick and simple visual summaries of data values. The data are divided into several groups, which are then displayed as a histogram consisting of a line of asterisks for each group. The number of asterisks in each line is proportional to the number of values assigned to that group; this figure is also printed at the beginning of each line. LPHISTOGRAM plots histograms in "line-printer" format, that is, using the characters of ordinary output rather than a high-resolution graphics display. It was added in Release 10 as a synonym of the earlier HISTOGRAM directive, which may become a command for high-resolution graphics in a future release.

The data for the histogram are specified using the DATA parameter in either variates, factors or one-way tables. If a histogram is to be formed from a variate, Genstat sorts its values into groups as defined by upper and lower bounds. You can also specify a list of variates, to obtain a parallel histogram. For each group one row of asterisks is printed for each variate, labelled by the corresponding identifier. The variates are sorted according to the same intervals; there is no need for them all to have the same numbers of values.

With variates of data, you can use the NGROUPS option to specify the number of groups in the histogram; Genstat will then work out appropriate limits, based on the range of the data, to form intervals of equal width. For example:

```
LPHISTOGRAM [NGROUPS=5] Data
```

Alternatively, you can define the groups explicitly, by setting the LIMITS option to a variate containing the group limits. For example:

```
VARIATE [VALUES=1,2,3,5,7,8,10] Glimits
LPHISTOGRAM [LIMITS=Glimits] Data
```

Glimits is a variate with seven values, producing a histogram in which the data are split into

eight groups; ≤1, 1-2, 2-3, 3-5, 5-7, 7-8, 8-10, >10. The upper limit of each group is included within that group, so the group 3-5, for example, contains values that are greater than 3 and less than or equal to 5. The values of the limits variate are sorted into ascending order if necessary, but the variate itself is not changed.

You can use the LABELS option to provide your own labelling for the groups of the histogram. It should be set to a text vector of length equal to the number of groups. If neither NGROUPS nor LIMITS has been set, the number of groups is determined from the number of values in the LABELS structure. If LABELS is also unset, the default number of groups is chosen as the integer value nearest to the square root of the number of values, up to a maximum of 10. Alternatively, procedure AKAIKEHISTOGRAM provides a more sophisticated method of generating histograms, using Akaike's Information Criterion (AIC) to generate an optimal grouping of the data.

The data for the histogram can also be specified as a factor (which defines the assignment of each unit to a group of the histogram). Genstat then counts the number of units that occur with each level of the factor; thus the number of groups of the histogram is the number of levels of the factor and the value for each group is the corresponding total. If the LABELS option is unset, the labels of the factor (if present) are used to label the groups, otherwise Genstat uses the factor levels.

When Genstat plots the histogram of a one-way table, the number of groups is the number of levels of the factor classifying the table and the values of the table indicate the number of observations in each group. If the LABELS option is unset, the labels or levels of the classifying factor are again used to label the histogram.

When producing a parallel histogram the data structures must all be of the same type: variate, factor or table. If parallel histograms are to be formed from several factors, they must all have the same number of levels, and the labels or levels of the first factor will be used to identify the groups. Likewise, if you are forming parallel histograms from several tables, they must all have the same number of values, and the classifying factor of the first table will define the labelling of the histogram.

The SYMBOLS parameter can specify alternative plotting characters to be used instead of the asterisk. For example:

```
LPHISTOGRAM Variate; SYMBOLS='+'
```

You can specify a different string for each structure in a parallel histogram. If you specify strings of more than one character, Genstat uses the characters in order, recycled as necessary, until each histogram bar is of the correct length.

The TITLE option lets you set an overall title for the output, and the DESCRIPTION parameter can be used to provide a text for labelling the histogram instead of the identifiers of the DATA structures.

Normally one asterisk will represent one unit. However, if there are many data values and the groups become large, Genstat may not be able to fit enough asterisks into one row. It will then alter the scaling so that one asterisk represents several units. You can set the scaling explicitly using the SCALE option; the value specified is rounded to the nearest integer, and determines how many units should be represented by each asterisk.

LPHISTOGRAM has two output parameters that allow you to save information that has been generated during formation of the histogram. The NOBSERVATIONS parameter allows you to save a one-way table of counts that contains the number of observations that were assigned to each group; the missing-value cell of this table will contain a count of the number of units that were missing and that therefore remain unclassified. When producing a histogram from a variate, you can use the GROUPS parameter to specify a factor to record the group to which each unit was allocated.

Normally, output goes to the current output channel, but you can use the CHANNEL option to direct it to another. For example, when you are working interactively, you might want to send a graph to a secondary output file so that you can print it later. Unlike some directives (for

example, PRINT) you cannot save the output in a text structure.

Options: CHANNEL, TITLE, LIMITS, NGROUPS, LABELS, SCALE.
Parameters: DATA, NOBSERVATIONS, GROUPS, SYMBOLS, DESCRIPTION.

Action with RESTRICT
You can restrict a DATA variate or factor to form a histogram for only a subset of the units. However, the restriction does not carry over to any other variates or factors listed by the DATA parameter.

See also
Directives: BARCHART, DHISTOGRAM, LPGRAPH, LPCONTOUR.
Genstat Reference Manual 1 Summary section on: Graphics.

LRV

Declares one or more LRV data structures.

Options

ROWS = *scalar*, *vector* or *pointer*	Number of rows, or row labels, for the matrix; default *
COLUMNS = *scalar*, *vector* or *pointer*	Number of columns, or column labels, for matrix and diagonal matrix; default *

Parameters

IDENTIFIER = *identifiers*	Identifiers of the LRVs
VECTORS = *matrices*	Matrix to contain the latent vectors for each LRV
ROOTS = *diagonal matrices*	Diagonal matrix to contain the latent roots for each LRV
TRACE = *scalars*	Trace of the matrix

Description

The LRV is a compound data structure. These are similar to pointers in that they point to other structures, but they have a fixed number of elements which must be of the correct types and must form a consistent set (in terms of their sizes and so on). You can refer to elements of compound structures in exactly the same way as the elements of pointers, but the suffixes and their labels are fixed for each type of structure. Unlike pointers, the labels are also not case sensitive; Genstat will recognize the label in either upper case or lower case, or in any mixture of the two.

The LRV structure is used to store latent roots and vectors resulting from the decomposition of a matrix (by the FLRV directive), or produced in multivariate analysis. It points to three structures (identified by their suffixes):

[1] or ['Vectors']	is a matrix whose columns are the latent vectors (the word "Vector" is used here in its mathematical sense rather than in the more specific Genstat sense; in fact, latent vectors are most conveniently stored in matrices rather than in Genstat vectors);
[2] or ['Roots']	is a diagonal matrix whose elements are the latent roots;
[3] or ['Trace']	is a scalar holding the trace of the matrix, which is the sum of all its latent roots.

The length of each latent vector is specified by the ROWS option; this then defines the number of rows in the 'VECTORS' matrix. The COLUMNS option defines the number of latent roots to be stored; this is also the number of latent vectors, and so indicates the number of columns in the 'VECTORS' matrix and the number of elements in the 'ROOTS' matrix. If you do not specify the number of columns Genstat will set it to be the same as the number of rows. The value of COLUMNS can be less than the value of ROWS; however, it must not exceed than that of ROWS, otherwise Genstat gives an error diagnostic. Row and column labels can be defined, as in the MATRIX directive.

You can specify identifiers for the three individual elements of the LRV by using the VECTORS, ROOTS and TRACE parameters. If you have declared them already they must be of the correct type (and you can also have given them values). If you have given these identifiers row or column settings, then these will be used for the LRV declaration and must match any of the corresponding options of LRV that you choose to set.

Options: ROWS, COLUMNS.
Parameters: IDENTIFIER, VECTORS, ROOTS, TRACE.

See also

Directives: FLRV, DIAGONALMATRIX, MATRIX, POINTER, QRD, SVD, SYMMETRICMATRIX.

Genstat Reference Manual 1 Summary sections on: Data structures, Multivariate and cluster analysis.

MARGIN

Forms and calculates marginal values for tables.

Option

CLASSIFICATION = *factors*	Factors classifying the margins to be formed; default * requests all margins to be formed

Parameters

OLDTABLE = *tables*	Tables from which the margins are to be taken or calculated
NEWTABLE = *tables*	New tables formed with margins
METHOD = *string tokens*	Way in which the margins are to be formed for each table (totals, means, minima, maxima, variances, medians, deletion, or a null string to indicate that the marginal values are all to be set to the missing value); default tota

Description

You can use MARGIN to extend a table to contain marginal values, or to change the marginal values of a table that already has margins, or to delete the margins from a table. The tables whose margins are to be changed are specified by the OLDTABLE parameter. If you specify only this parameter, the new values replace those of the original tables. However, if you want to retain the original values, you can specify new tables to contain the amended values, using the NEWTABLE list. These tables will be declared automatically, if you have not declared them already.

The METHOD parameter controls the type of margins that are formed. If you set METHOD=deletion, all the margins of the tables are deleted but the body of the table is retained.

The CLASSIFICATION option specifies the list of factors for which you want to form marginal values. Genstat puts missing values in the margins that are excluded if the METHOD parameter is set to maxima or minima; for other settings of METHOD, Genstat puts in zeroes. The classifying sets for each table can be different, but all the factors in the CLASSIFICATION option must be in the classifying sets of each OLDTABLE.

Option: CLASSIFICATION.
Parameters: OLDTABLE, NEWTABLE, METHOD.

See also

Directives: TABLE, TABULATE, COMBINE.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

MATRIX

Declares one or more matrix data structures.

Options

ROWS = *scalar*, *vector*, *pointer* or *text*	Number of rows, or labels for rows; default *
COLUMNS = *scalar*, *vector*, *pointer* or *text*	Number of columns, or labels for columns; default *
VALUES = *numbers*	Values for all the matrices; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the matrices in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the matrices
VALUES = *identifiers*	Values for each matrix
DECIMALS = *scalars*	Number of decimal places for printing
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for the contents of each structure
MAXIMUM = *scalars*	Maximum value for the contents of each structure
DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represent dates and times

Description

A matrix stores a set of numbers as a two-dimensional array indexed by rows and columns. For example, the array

```
 1   2   3   4
 5   6   7   8
 9  10  11  12
```

is called a three-by-four matrix.

You use the ROWS and COLUMNS options to specify the size of the matrices that are being defined. The simplest way of doing this is to use scalars to define the numbers of rows and columns explicitly. Alternatively, you can set ROWS (or COLUMNS) to a variate, text or pointer, whose length then defines the number of rows (or columns) and whose values will then be used as labels, for example when the matrix is printed. Finally, if you specify a factor, the number of levels defines the number of rows or columns and the labels if available, or otherwise the levels, are used for labelling.

Values can be supplied for the matrices using either the VALUES option or the VALUES parameter. The option defines a common value (or set of values) for all the matrices in the declaration, while the parameter allows them each to be given different values. With the option you must supply a list of values. With the parameter, however, you must give a list of identifiers of data structures of the appropriate mode; unnamed data structures are particularly useful for this. Thus, to declare the matrix above, we can put:

```
MATRIX [ROWS=3; COLUMNS=4] X; \
  VALUES=!(1,2,3,4,5,6,7,8,9,10,11,12)
```

If both the option and the parameter are specified, the parameter takes precedence.

The DECIMALS parameter can be used to define the number of decimal places that Genstat will

use by default whenever the values of the matrix are printed. This applies to output either by PRINT or from an analysis (but it does not affect the accuracy with which the numbers are stored).

You can associate a text with each data structure by means of the parameter EXTRA. This text is then used by many Genstat directives to give a fuller annotation of output.

The MINIMUM and MAXIMUM parameters allow you to define lower and upper limits on the values expected for any structure that stores numbers. Genstat then prints warnings if any values outside that range are assigned to the structure.

The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each matrix to indicate that the matrix stores dates and times, and to define a format to be used for these, by default, when they are printed; details are given in the description of the PRINT directive.

If you are declaring any of the matrices for a second time, by default you will lose all its existing attributes and values. You can retain those that remain valid by setting option MODIFY=yes.

The IPRINT option can be set to specify how the matrices will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

Options: ROWS, COLUMNS, VALUES, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUES, DECIMALS, EXTRA, MINIMUM, MAXIMUM, DREPRESENTATION.

See also
Directives: DIAGONALMATRIX, LRV, SYMMETRICMATRIX, SSPM.
Genstat Reference Manual 1 Summary section on: Data structures.

MCOVARIOGRAM

Fits models to sets of variograms and cross-variograms.

Options

PRINT = *string tokens*	Controls printed output from the fit (model, summary, estimates, fittedvalues, monitoring); default mode, summ, esti
WEIGHTING = *string token*	Method to be used for weighting (counts, equal); default coun
MAXLAG = *scalar*	Maximum lag distance of points to be included in the modelling
MINCOUNT = *scalar*	Minimum number of points required at a particular lag point for a pair of variables for this to be used to model their cross-variogram; default 30 for equal weighting and 10 for counts
MAXCYCLE = scalar	Maximum number of iterations for model fitting; default 30
TOLERANCES = *variate*	Tolerances for model fitting; default * i.e. appropriate default values
COORDSYSTEM = *string token*	Coordinate system used for the geometry for discretizing the lag (mathematical, geographical); default math
COVARIOGRAM = *pointers*	Experimental variograms, cross-variograms and associated information defining the data for fitting the model

Parameters

MODELTYPE = *string tokens*	Defines the model structures to be fitted (nugget, power, boundedlinear, circular, spherical, pentaspherical, cubic, stable, besselk1, cardinalsine, dampenedcosine); no default i.e. must be specified
INITIAL = *scalars* or *variates*	Scalar defining the initial distance parameter for fitting an isotropic model structure or a variate defining initial values for an anisotropic ellipse or ellipsoid for fitting an geometrical anisotropic model
ISOTROPY = *string tokens*	Specifies the zonal anisotropy to be used for model structure (isotropic, x, y, z, xy, xz, yz); default isot
ESTIMATES = *pointers*	Structures to store the estimated nonlinear parameters and sill values
LOWER = *scalars*	Lower bound for each nonlinear distance parameter
UPPER = *scalars*	Upper bound for each nonlinear distance parameter
STEPLENGTH = *scalars*	Initial step length for each nonlinear distance parameter
SMOOTHNESS = *scalars*	Value of exponent parameter for the power and stable models, or theta parameter for the dampened-cosine model

Description

The MCOVARIOGRAM directive fits models to sets of auto- and cross-variograms. You can specify a combination of basic variogram functions to model the variograms, for example, nugget plus spherical. MCOVARIOGRAM uses the algorithms from the directives FIT and FITNONLINEAR to

estimate the model parameters for the combination of basic variogram functions. It then fits a linear model of coregionalization using the Goulard & Voltz (1992) algorithm, where each step of the solution is checked for conditional semi-definiteness. The two-step process is iterated until convergence.

The MODELTYPE parameter selects the combination of model structures to be used in the model:

nugget	c_0
boundlinear	ch/a for h ≤ a, otherwise 0
circular	$c\{1 - (2/\pi)\arccos(h/a) + (2h/(\pi a))\sqrt{(1-h^2/a^2)}\}$ for h ≤ a, otherwise 0
spherical	$c\{1.5h/a - 0.5(h/a)^3\}$ for h ≤ a, otherwise 0
pentaspherical	$c\{1.875h/a - 1.25(h/a)^3 + 0.375(h/a)^5\}$ for h ≤ a, otherwise 0
cubic	$c\{7(h/a)^2 - 8.75(h/a)^3 + 3.5(h/a)^5 - 0.75(h/a)^7\}$
stable	$c\{1 - \exp(-(h/a)^b))\}$ for $0 \le b \le 2$
besselk1	$c\{1 - h/a\, k_1(h/a)\}$
cardinalsine	$c\{1 - a/h \sin(h/a)\}$
dampenedcosine	$c\{1 - \exp(-h/(as))\cos(h/a)\}$
power	gh^α

Initial values for the model structures should be supplied using the INITIAL parameter. For an isotropic model the initial value should be specified as a scalar. You can specify a geometrically anisotropic model by supplying the values within a variate. In two dimensions the variate should contain three values that define an anisotropy ellipse. The first value should define the first axis direction. This is the angle for the main direction of continuity (least change with separating distance) measured in degrees, counter-clockwise from East if option COORDSYSTEM is set to mathematical or clockwise from North if COORDSYSTEM is set to geographical. The second value should contain the initial value for the distance parameter of the first axis, and the last value of the variate should be the anisotropy ratio between the distance parameters along the first axis (principal direction of continuity) and the second axis.

In three dimensions the variate should contain six values that define an anisotropy ellipsoid. The first value defines the angle for the first axis (principal direction of continuity) which is measured in degrees, counter-clockwise from East if COORDSYSTEM is set to mathematical or clockwise from North if COORDSYSTEM is set to geographical. The second value defines the dip angle for the first axis (rotation angle around the y-axis) which is measured in degrees up from horizontal. The third value defines the rotation angle of the second and third axis around the first axis (defined by the two previous angles). The fourth value should contain the initial value for the distance parameter along the first axis. The fifth value defines the anisotropy ratio between distance parameters along the first and second axis of the ellipsoid. The last value of the variate defines the anisotropy ratio between the distance parameters along the second and third axis of the ellipsoid.

Another form of anisotropy can occur when the sill of a semi-variogram varies in different directions. This is known as zonal anisotropy and you can set a model structure to be zonal in particular directions using the ISOTROPY parameter. A model structure can be zonal and geometrically anisotropic.

For the power and stable models the SMOOTHNESS option controls the power parameter for the model. By default, the parameter is estimated, however, you can supply a value to fix the parameter for the model fitting.

The WEIGHTING option controls the weights that are used when fitting the model. The default setting counts uses the values supplied for the counts within the COVARIOGRAM option, and

`equal` uses equal weights (of one).

The `MAXLAG` option can be used to specify the maximum lag distance of points to be included in the modelling. The `MINCOUNT` option specifies the minimum number of points to be used to model the variograms at a particular lag.

The `TOLERANCES` option controls the criterion for convergence of the nonlinear regression and Goulard & Voltz algorithm. The values should be supplied in a variate where the first value is the criterion for the nonlinear regression and the second value is the criterion for the Goulard & Voltz algorithm. The option `MAXCYCLE` can be used to change the maximum number of iterations performed by the nonlinear regression from the default of 30.

The `COVARIOGRAM` option allows you to specify a pointer containing the auto-variograms, cross-variograms and associated information. This structure can be saved from the `FCOVARIOGRAM` directive.

The geometry used for the directions supplied using the `COVARIOGRAM` option is given by the `COORDSYSTEM` option, where the setting `mathematical` specifies directions counter-clockwise from East, and `geographical` clockwise from North (for the first angle only in 3 dimensions).

The `ESTIMATES` parameter allows you to specify an identifier to save the estimated nonlinear parameters, sill values and associated information. This structure stores the information required for the `DCOVARIOGRAM` procedure or `COKRIGE` directive.

The `PRINT` option controls the output to be displayed, with settings:

`model`	description of the models fitted,
`summary`	summary of analysis,
`estimates`	parameter estimates,
`fittedvalues`	fitted semi-variances,
`monitoring`	monitoring information at each iteration of the nonlinear regression.

Options: `PRINT`, `WEIGHTING`, `MAXLAG`, `MINCOUNT`, `MAXCYCLE`, `TOLERANCES`, `COORDSYSTEM`, `COVARIOGRAM`.

Parameters: `MODELTYPE`, `INITIAL`, `ISOTROPY`, `ESTIMATES`, `LOWER`, `UPPER`, `STEPLENGTH`, `SMOOTHNESS`.

Reference

Goulard, M. & Voltz, M. (1992). Linear coregionalization model: tools for estimation and choice of cross-variogram matrix. *Mathematical Geology*, **24**, 269-286.

See also

Directives: `FCOVARIOGRAM`, `COKRIGE`, `FVARIOGRAM`, `KRIGE`.

Procedures: `DCOVARIOGRAM`, `KCROSSVALIDATION`, `MVARIOGRAM`, `DVARIOGRAM` `DHSCATTERGRAM`.

Genstat Reference Manual 1 Summary section on: Spatial statistics.

MDS

Performs non-metric multidimensional scaling.

Options

PRINT = *string tokens*	Printed output required (coordinates, roots, distances, fitteddistances, stress, monitoring); default * i.e. no printing
DATA = *symmetric matrix*	Distances amongst a set of units
METHOD = *string token*	Whether to use non-metric scaling, or metric scaling with linear regression of the fitted distances to the actual distances (nonmetric, linear); default nonm
SCALING = *string token*	Whether least-squares, least-squares-squared, or log-stress scaling is to be used (ls, lss, logstress); default ls
TIES = *string token*	Treatment of tied data values (primary, secondary, tertiary); default prim
WEIGHTS = *symmetric matrix*	Weights for each distance value; default * i.e. all distances with weight one
INITIAL = *matrix*	Initial configuration; default * i.e. a principal coordinate solution is used
NSTARTS = *scalar*	Number of starting configurations to be used, by making random perturbations to the initial configuration; default 10
SEED = *scalar*	Seed for the random-number generator; default 0
MAXCYCLE = *scalar*	Maximum number of iterations; default 30

Parameters

NDIMENSIONS = *scalars*	Number of dimensions for each solution
COORDINATES = *matrices*	To store the coordinates of the units for each solution
STRESS = *scalars*	To store the stress value for each solution
DISTANCES = *symmetric matrices*	To store the distances amongst the points for the units in the fitted number of dimensions
FITTEDDISTANCES = *symmetric matrices*	To store the fitted distances from the monotonic (METHOD=nonmetric) or linear (METHOD=linear) regression

Description

The MDS directive carries out iterative scaling, including metric and non-metric scaling. The input data consists of a symmetric matrix whose values may be interpreted, in a general sense, as distances between a set of objects. The matrix is specified by the DATA option; thus only one matrix can be analysed each time the MDS directive is used.

The objective of the MDS directive is to find a set of coordinates whose inter-point distances match, as closely as possible, those of the input data matrix. When plotted, the coordinates provide a display which can be interpreted in the same way as a map: for example, if points in the display are close together, their distance apart in the data matrix was small.

The algorithm invoked by the MDS directive uses the method of steepest descent to guide the algorithm from an initial configuration of points to the final matrix of coordinates that has the minimum stress of all configurations examined.

Printed output is controlled by the PRINT option; by default nothing is printed. There are six possible settings:

`coordinates`	prints the solution coordinates, rotated to principal coordinates;
`roots`	prints the latent roots of the solution coordinates;
`distances`	prints the inter-unit distances, computed from the solution configuration;
`fitteddistances`	prints the fitted values from the regression of the inter-unit distances on the distances in the data matrix, the regression may be monotonic or linear through the origin, depending on the setting of the `METHOD` option;
`stress`	prints the stress of the solution coordinates;
`monitoring`	prints a summary of the results at each iteration.

The `METHOD` option determines whether metric or non-metric scaling is given. The algorithm involves regression of the distances, calculated from the solution coordinates, against the dissimilarities in the symmetric matrix specified by the `DATA` option. With the default setting, `METHOD=nonmetric`, monotonic regression is used; if `METHOD=linear`, the algorithm uses linear regression through the origin.

The stress function to be minimized can be selected using the `STRESS` option. There are three possibilities.

`ls` (least squares):	$\sqrt{\{ \sum_i \sum_j \{w_{ij} (d_{ij} - \hat{d}_{ij})^2\} / (m \sum_i \sum_j \{ w_{ij} d_{ij}^2\})\}}$
`lss` (least-squares-squared):	$\sqrt{\{ \sum_i \sum_j \{w_{ij} (d_{ij}^2 - \hat{d}_{ij}^2)^2\} / (m \sum_i \sum_j \{ w_{ij} d_{ij}^4\})\}}$
`logstress`:	$\sqrt{\{ \sum_i \sum_j \{w_{ij} (\log(d_{ij}) - \log(\hat{d}_{ij}))^2\} / m \}}$

where the d_{ij} are the elements of the dissimilarity matrix calculated for the fitted configuration, the $\hat{d}_{ij}$ are the fitted values from the regression selected by the `METHOD` option, the w_{ij} are the corresponding weights and m is the number of off-diagonal elements in the dissimilarity matrix.

The `TIES` option allows you to vary the way in which tied data values in the input data matrix are to be treated. By default, the treatment of ties is primary, and no restrictions are placed on the distances corresponding to tied dissimilarities in the input data matrix. In the secondary treatment of ties, the distances corresponding to tied dissimilarities are required to be as nearly equal as possible. Kendall (1977) describes a compromise between the primary and secondary approaches to ties: the block of ties corresponding to the smallest dissimilarity are handled by the secondary treatment, the remaining blocks of ties are handled by the primary treatment. This tertiary treatment of ties is useful when the dissimilarities take only a few values. For example, in the reconstruction of maps from abuttal information, the dissimilarity coefficient takes only two values: zero if localities abut, and one if they do not. The block of ties associated with the dissimilarity of zero are handled by the secondary treatment, and the block of ties with dissimilarity one by the primary treatment.

The `WEIGHT` option can be used to specify a symmetric matrix of weights. Each element of the matrix gives the weight to be attached to the corresponding element of the input data matrix. If the option is not set, the elements of the data matrix are weighted equally: w_{ij}=1 for all i and j. The most important use of the option occurs when the matrix of weights contains only zeros and ones; the zeros then correspond to missing values in the input data matrix, allowing incomplete data matrices to be scaled. Up to about two thirds of the data matrix may be missing before the algorithm breaks down. This enables experimenters to design studies in which only a subset of all the dissimilarities need to be observed. This is particularly useful when there are a large number of units; if the number of units is m, say, a complete $m \times m$ data matrix requires $m(m-1)/2$ dissimilarities to be observed.

Since the algorithm is an iterative one, making use of the method of steepest descent, there is no guarantee that the solution coordinates found from any given starting configuration has the minimum stress of all possible configurations. The algorithm may have found a local, rather than the global, minimum. This problem may be partially overcome by using a series of different starting configurations. If several of the solutions arrive at the same lowest stress solution, then

you may be reasonably confident of having found the global minimum. The NSTARTS option determines the number of starting configurations to be used. The starting configuration used on the first start can be specified by the INITIAL option; if this is not set, the default is to take the principal coordinate solution obtained from a PCO analysis of the input dissimilarity matrix. Subsequent starting configurations are found by perturbing each coordinate of the first starting configuration by successively larger amounts. This strategy generally results in at least one starting configuration that does not get entrapped in a local minimum: however there can be no guarantee that the global minimum for the stress function has been found. Experience suggests that, for safety, the NSTARTS option should be set equal to at least 10. By default NSTARTS=10.

The SEED option supplies the seed for the random numbers that are used to perturb the initial configuration. The default of zero continues the existing sequence of random numbers if MDS has already been used in the current Genstat job. If MDS has not yet been used, Genstat picks a seed at random.

The MAXCYCLES option determines the maximum number of iterations of the algorithm. The default of 30 should usually be sufficient. However, it may be necessary to set a larger value for very large data matrices or when using the logstress setting of the SCALING option. The monitoring setting of the PRINT option may be used to see how convergence is progressing.

The NDIMENSIONS parameter must be set to a scalar (or scalars) to indicate the number(s) of dimensions in which the multidimensional scaling is to be performed on the data matrix. An MDS statement with a list of scalars will carry out a series of scaling operations, all based on the same matrix of dissimilarities, but with different numbers of dimensions.

The remaining parameters of the MDS directive allow output to be saved in Genstat data structures. The COORDINATES parameter can list matrices to store the minimum stress coordinates in each of the dimensions given by the NDIMENSIONS parameter, and the STRESS parameter can specify scalars to store the associated minimum stresses. The parameters DISTANCES and FITTEDDISTANCES can specify symmetric matrices to store the distances computed from the coordinates matrix and the fitted distances computed from the monotonic or linear regressions, respectively.

Options: PRINT, DATA, METHOD, SCALING, TIES, WEIGHTS, INITIAL, NSTARTS, SEED, MAXCYCLE.
Parameters: NDIMENSIONS, COORDINATES, STRESS, DISTANCES, FITTEDDISTANCES.

Reference

Kendall, D.G. (1977). On the tertiary treatment of ties. *Proceedings of the Royal Society of London, Series A*, **354**, 407-423.

See also

Directives: MONOTONIC, PCP, PCO, CVA, FCA.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

MERGE

Copies subfiles from backing-store files into a single file.

Options

PRINT = *string token*	What to print (catalogue); default *
OUTCHANNEL = *scalar*	Channel number of the backing-store file where the subfiles are to be stored; default 0, i.e. the workfile
METHOD = *string token*	How to append subfiles to the OUT file (add, overwrite, replace); default add, i.e. clashes in subfile identifiers cause a fault (note: replace overwrites the complete file)
PASSWORD = *text*	Password to be checked against that stored with the file; default *

Parameters

SUBFILE = *identifiers*	Identifiers of the subfiles
INCHANNEL = *scalars*	Channel number of the backing-store file containing each subfile
NEWSUBFILE = *identifiers*	Identifier to be used for each subfile in the new file

Description

The MERGE directive is used to copy subfiles into another backing-store file. You can either add the subfiles to an existing backing-store file, or form a new backing-store file.

The OUTCHANNEL option specifies the backing-store channel of the file to which the subfiles are to be copied; by default this is the workfile (channel 0).

The SUBFILE parameter specifies the list of subfiles that are to be copied, and the INCHANNEL parameter indicates the channel of the backing-store file where each one is currently stored. If you do not specify the INCHANNEL parameter, Genstat assumes that the subfiles are coming from the workfile. You are not allowed to include the OUTCHANNEL among the channels in the INCHANNEL list. Also, you cannot store two subfiles with the same names, and should use the NEWSUBFILE parameter to rename any that clash. For example

```
MERGE [OUTCHANNEL=3] JanData,JulyData,JanData; \
  INCHANNEL=1,1,2; NEWSUBFILE=Jan92dat,Jul92dat,Jan93dat
```

To rename only some of the subfiles, you can either respecify the existing identifier, or insert * at the appropriate point in the NEWSUBFILE list.

If you specify a missing identifier * in the SUBFILE list, Genstat will include all the subfiles from the relevant INCHANNEL. If you want to rename any of these subfiles, you can also mention it explicitly. For example, this statement will take all the subfiles from channel 1 and rename subfile Sub as Subf.

```
MERGE *,Sub; INCHANNEL=1; NEWSUBFILE=*,Subf
```

You can set option PRINT=catalogue to produce a catalogue of the subfiles in the new backing-store file.

If a subfile of the specified name already exists on the backing-store file, the storing operation will usually fail. However, you can set option METHOD=overwrite to overwrite the old subfile, that is, to replace the old subfile with a new subfile. Alternatively, you can put METHOD=replace to form a new backing-store file containing only the new subfiles.

Subfiles are merged in a fixed order. Genstat first takes the subfiles from the backing-store file with the lowest channel number, in the order in which they occur there, then it takes the subfiles the next lowest channel number, and so on. If OUTCHANNEL=0 (that is, the new file is the workfile), the original subfiles that are to be retained from that file will be followed by the new subfiles; otherwise, if OUTCHANNEL is non-zero, the original subfiles are placed after the new

subfiles. If you want to put the subfiles into a particular order, you should merge them into the workfile in that order, and then merge the workfile into a new userfile.

To keep the new file secure, you can use the PASSWORD option to incorporate a password. Once you have done this, you must include the same password in any future use of MERGE or STORE with this same userfile; spaces, case, and newlines are significant in the password. You cannot change the password in a userfile once you have set it, but you can use the MERGE directive to create a new userfile with no password or with a new password. If you set the password to be a text whose values have been have restricted, the restriction is ignored.

Options: PRINT, OUTCHANNEL, METHOD, PASSWORD.
Parameters: SUBFILE, INCHANNEL, NEWSUBFILE.

See also
Directives: STORE, RETRIEVE, CATALOGUE, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

MODEL

Defines the response variate(s) and the type of model to be fitted for linear, generalized linear, generalized additive and nonlinear models.

Options

DISTRIBUTION = *string token*	Distribution of the response variable (normal, poisson, binomial, gamma, inversenormal, multinomial, calculated, negativebinomial, geometric, exponential, bernoulli); default norm
LINK = *string token*	Link function (canonical, identity, logarithm, logit, reciprocal, power, squareroot, probit, complementaryloglog, calculated, logratio); default cano (i.e. iden for DIST=norm or calc; loga for DIST=pois; logi for DIST=bino, bern or mult; reci for DIST=gamm or expo; powe for DIST=inve; logr for DIST=nega or geom)
EXPONENT = *scalar*	Exponent for power link; default -2
AGGREGATION = *scalar*	Fixed parameter for negative binomial distribution (parameter k as in variance function Var = mean + mean$^2/k$); default 1
KLOGRATIO = *scalar*	Parameter for logratio link, in form log(mean/(mean+k)); default as set in AGGREGATION option
DISPERSION = *scalar*	Value of dispersion parameter in calculation of s.e.s etc; default * for DIST=norm, gamm, inve or calc, and 1 for DIST=pois, bino, mult, nega, geom, expo or bern
WEIGHTS = *variate* or *symmetric matrix*	Variate of weights for weighted regression, or symmetric matrix of weights (one row and column for each unit of data) for generalized least squares; default *
OFFSET = *variate*	Offset variate to be included in model; default *
GROUPS = *factor*	Absorbing factor defining the groups for within-groups linear or generalized linear regression; default *
RMETHOD = *string token*	Type of residuals to form, if any, after each model is fitted (deviance, Pearson, simple); default devi
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default devi
FUNCTIONVALUE = *scalar*	Scalar whose value is to be minimized by calculation; default *
YRELATION = *string token*	Whether to analyse the y-variates separately, as in ordinary regression, or to analyse them cumulatively as counts in successive categories of a multinomial distribution (separate, cumulative); default sepa
DCALCULATION = *expression structures*	Calculations to define the deviance contributions and variance function for a non-standard distribution; must be specified when DIST=calc
LCALCULATION = *expression structures*	Calculations to define the fitted values and link derivative for a non-standard link; must be specified

	when `LINK=calc`
`DFDISPERSION` = *scalar*	Allows you to specify the number of degrees of freedom for a dispersion parameter specified by the `DISPERSION` option; if this is not set, the supplied dispersion is assumed to be known exactly
`SAVE` = *identifier*	To name regression save structure; default `*`

Parameters

`Y` = *variates*	Response variates; only the first is used in nonlinear models and in generalized linear models except when `DIST=mult`, when they specify the numbers in each category of an ordinal response model
`NBINOMIAL` = *variate* or *scalar*	Total numbers for `DIST=bino`
`RESIDUALS` = *variates*	To save residuals for each y variate after fitting a model
`FITTEDVALUES` = *variates*	To save fitted values, and provide fitted values if no terms are given in `FITNONLINEAR`
`LINEARPREDICTOR` = *variate*	Specifies the identifier of the variate to hold the linear predictor
`DERIVATIVE` = *variate*	Specifies the identifier of the variate to hold the derivative of the link function at each unit
`DEVIANCE` = *variate*	Specifies the identifier of the variate to hold the contribution to the deviance from each unit
`VFUNCTION` = *variate*	Specifies the identifier of the variate to hold the value of the variance function at each unit

Description

The `MODEL` directive does not actually fit anything: it simply sets up some structures inside Genstat that are used when you give a `FIT`, `FITCURVE` or `FITNONLINEAR` statement later on. So when you are doing regression, `MODEL` will always be accompanied by at least one other regression statement to fit a model, like `FIT`.

The `Y` parameter allows a list of variates; if you put more than one for linear regression, then you will get an analysis for each. This is a more efficient way of doing many linear regressions with the same explanatory variables, than separate pairs of `MODEL` and `FIT` statements. With additive models, generalized linear models and nonlinear models, only the first variate will be analysed (with the exception of multinomial response models); the others will be ignored.

The `RESIDUALS` and `FITTEDVALUES` parameters allow you to specify variates to contain the residuals and fitted values for each response variable. The residuals are the "unexplained" component of the response variable, standardized in some way according to the `RMETHOD` option. The fitted values are the "explained" component: that is, the combination of parameters and explanatory variables fitted in the model. You can get access to these sets of values in a different way through the `RKEEP` directive.

The `DISTRIBUTION` and `LINK` options are used to specify a *generalized linear model* (McCullagh & Nelder 1989). By default the data are assumed to follow a Normal distribution, as required for ordinary linear regression, but other distributions can be selected using the `DISTRIBUTION` option. The `LINK` option specifies the *link function* that relates the linear model to the expected values of the distribution; in the default ordinary linear regression, this is the identity function (indicating no transformation). So, for example, for a log-linear model we would specify `DISTRIBUTION=Poisson` and `LINK=log`, while for logistic regression we would have `DISTRIBUTION=binomial` and `LINK=logit`. The `NBINOMIAL` parameter must also be set when `DISTRIBUTION=binomial`, to give the number of binomial trials for each unit.

The `EXPONENT` option specifies the exponent when `LINK=power`. Similarly, the

AGGREGATION option specifies the aggregation parameter k when DISTRIBUTION=negativebinomial. This is a measure of the tendency for observations to cluster together which appears in the formula for the variance as a function of the mean

$$\text{variance} = \text{mean} + \text{mean}^2/k$$

The default value of k is set at 1, which corresponds to the geometric distribution. The parameter k must be positive, and as it increases to infinity the distribution approaches the Poisson distribution. The KLOGRATIO option sets the parameter k for the logratio link.

You can also define your own distribution or link function for a generalized linear model. To specify your own distribution, you need to set DISTRIBUTION=calculated and then specify expression structures with the DCALCULATION option to calculate the deviance and the variance function for each unit of the response variate, using the current values of the fitted-values variate. You must also set the FITTEDVALUES, DEVIANCE and VFUNCTION parameters to indicate which identifiers are used to represent these in the expressions. To specify your own link, you need to set LINK=calculated and provide expressions with the LCALCULATION option for two other calculations to form the fitted values and the derivative of the link function for each unit of the response variate, using the current values of the linear predictor. You must also set the FITTEDVALUES, LINEARPREDICTOR and DERIVATIVE parameters to specify the identifiers used to represent these in the calculations. In addition, you must provide initial values for the linear predictor, so that the iterative process can get started: often this can be done just by applying the link function to the response variate itself, but it may be necessary to modify extreme values such as 0 that may be mapped to infinity by the link function.

You can fit ordinal response models by setting option YRELATION=cumulative and option DISTRIBUTION=multinomial.

The DISPERSION option controls how the variance of the distribution of the response values is calculated. By default, the variance is estimated from the residual mean square, and standard errors and standardized residuals are calculated from the estimate. If you use DISPERSION to supply a value for the variance of the Normal distribution, or for the dispersion parameter of other distributions, then standard errors and residuals are based on this given value instead. In a generalized linear model, the dispersion of the chosen distribution can be fixed at a value provided by the DISPERSION option, or estimated from either the residual deviance or the Pearson chi-square statistic, as specified by the DMETHOD option.

The DFDISPERSION option allows you to specify the number of degrees of freedom for a value specified by the DISPERSION option. You might want to use this, for example, if you had estimated the dispersion from some other data set. If DFDISPERSION is not set, the supplied dispersion is assumed to be known exactly.

The WEIGHTS option allows you to specify a variate holding weights for each unit. In simple linear regression, the estimate of dispersion is then the weighted residual mean square. Thus, if the variance of the response variable is not constant, and you know the relative size of the variance for each observation, you can set the weight to be proportional to the inverse of the variance of an observation. Alternatively, if the variance is related in a simple way to the mean, you may just need to specify a different distribution for the response. The WEIGHTS option can also be set to a symmetric matrix, supplying weights corresponding to some pattern of correlation or covariance between units as well as variance of each unit. The subsequent analysis is known as generalized least-squares if the response distribution is Normal.

The OFFSET option allows you to include in the regression a variable with no corresponding parameter. Linear regression analysis of Y with offset O is just the same as analysis of $Y-O$, but the offset has non-trivial applications in generalized linear models.

The GROUPS option specifies a factor whose effects you want to eliminate before any regression is fitted. The factor must already have been defined. This method of elimination is sometimes called *absorption*; you might want to use it when data from many different groups are to be modelled. Use of GROUPS gives less information than you would get if you included

the factor explicitly in the model (leverages, predictions and some parameter correlations cannot be formed), but it saves space and time in fitting the model when the factor has many levels. You can use GROUPS only with linear and generalized linear regression.

The RMETHOD option controls how residuals are formed. By default, residuals are *deviance residuals* standardized by their estimated variance. The alternative *Pearson residuals* are defined in exactly the same way if the distribution is Normal, but for regression models with distributions other than Normal the two kinds of residual are different. If you do not want residuals, you can set the option to a missing value (*) to save space within Genstat. However, you will then not be able to get residuals, fitted values or leverages, and the automatic checks on the fit of a model will not be done.

The FUNCTIONVALUE option is relevant only when you want to use FITNONLINEAR to optimize a general function. It then identifies the scalar that stores the results in the expression that calculates the function to be minimized (see the CALCULATION option of FITNONLINEAR). This should calculate a deviance if you are using this general facility to fit a statistical model. FUNCTIONVALUE is ignored if the Y parameter of MODEL is set.

The SAVE option allows you to specify an identifier for the regression save structure. This structure stores the current state of the regression model, and can be used explicitly in the directives RDISPLAY, RKEEP, PREDICT and RFUNCTION. If the identifier in SAVE is of a regression save structure that already has values, those values are deleted. You can reset the current regression save structure at any point in a program by using the SET directive. Then, later regression statements would use the model stored in this save structure.

Options: DISTRIBUTION, LINK, EXPONENT, AGGREGATION, KLOGRATIO, DISPERSION, WEIGHTS, OFFSET, GROUPS, RMETHOD, DMETHOD, FUNCTIONVALUE, YRELATION, DCALCULATION, LCALCULATION, DFDISPERSION, SAVE.

Parameters: Y, NBINOMIAL, RESIDUALS, FITTEDVALUES, LINEARPREDICTOR, DERIVATIVE, DEVIANCE, VFUNCTION.

Action with RESTRICT

You can restrict the units that Genstat will use for the regression by putting a restriction on any of the vectors involved in the MODEL statement (response variates, weight variate, offset variate, grouping factor or variate of binomial totals), or on any explanatory variate or factor in a subsequent TERMS statement. However, you are not allowed to have different restrictions on the different vectors. You should not alter the restriction applied to the vectors between the TERMS statement and subsequent fitting statements.

Reference

McCullagh, P. & Nelder, J.A. (1989). *Generalized Linear Models* (second edition). Chapman and Hall, London.

See also

Directives: FIT, FITCURVE, FITNONLINEAR, TERMS.

Genstat Reference Manual 1 Summary section on: Regression analysis.

MONOTONIC

Fits an increasing monotonic regression of y on x.

No options

Parameters

`Y` = *variates*	Y-values of the data points
`X` = *variates*	X-values of the data points; default is to assume that the x-values are monotonically increasing
`RESIDUALS` = *variates*	Variate to save the residuals from each fit
`FITTEDVALUES` = *variates*	Variate to save the fitted values from each fit

Description

Monotonic regression plays a key role in non-metric multidimensional scaling, which is available in Genstat via the `MDS` directive. However, it can be useful in its own right, so the method has been made accessible by the `MONOTONIC` directive. A monotonic regression through a set of points is simply the line that best fits the points subject to the constraint that it never decreases: of course the line need not be straight, in fact it rarely will be. If you need a monotonically decreasing line, you can simply subtract all the y-values from their maximum, find the monotonically increasing regression, and then back-transform the data and fitted line, and change the sign of the residuals.

The `MONOTONIC` directive has no options. It has four parameters: `Y` to specify the y-values, `X` for the x-values, `RESIDUALS` to save the residuals, and `FITTEDVALUES` to save the fitted values. The x-values need not be supplied, in which case the directive assumes that the y-values are in increasing order of the x-values. In common with the other regression directives, the variates to save the residuals and fitted values need not be declared in advance.

Options: none.
Parameters: `Y`, `X`, `RESIDUALS`, `FITTEDVALUES`.

Action with RESTRICT

`MONOTONIC` ignores any restrictions on the variates.

See also

Directives: `MDS`, `FIT`, `FITCURVE`, `FITNONLINEAR`.
Genstat Reference Manual 1 Summary sections on: Multivariate and cluster analysis, Regression analysis.

NAG

Calls an algorithm from the NAG Library.

Options

PRINT = *string token*	Controls printed output (algorithms, monitoring); default * i.e. none
NAME = *string token*	Name of the algorithm to call; default * i.e. none
ZDZ = *string token*	Value to be given to zero divided by zero in Genstat expressions defined in the ARGUMENTS (missing, zero); default miss
TOLERANCE = *scalar*	If the scalar is non missing, this defines the smallest non-zero number for use in Genstat expressions defined in the ARGUMENTS; otherwise it accesses the default value, which is defined automatically for the computer concerned
SEED = *scalar*	Seed to use for any random number generation in Genstat expressions defined in the ARGUMENTS; default 0
INDEX = *scalar*	If a Genstat expression defined in the ARGUMENTS has a list of structures before the assignment operator (=), the scalar indicates the position within the list of the structure currently being evaluated

Parameters

ARGUMENTS = *pointer*	Arguments for the call
RESULT = *scalar*	Stores the result for algorithms that take the form of a function rather than a subroutine

Description

NAG provides access to some specific algorithms in the Numerical Algorithms Group's subroutine libraries. You can set option PRINT=algorithms to list those that are currently available. The other setting monitoring gives additional monitoring from algorithms like D02KDF that can give additional monitoring information from a MONIT subroutine. (NAG includes a custom version of MONIT for each routine, that provides all the relevant information.)

The name of the algorithm is specified using the NAME option. It is best to give the name in full, as the NAG names may not be distinct in their first four characters and so the standard abbreviation rules (e.g. that four characters are sufficient) cannot be guaranteed in all future releases. The arguments for the call are supplied, in a pointer, using the ARGUMENTS parameter. These must be in the order required by the algorithm, and input arguments must be of the correct type (number or string) and shape (vector, matrix and so on); for details see the relevant NAG documentation. Output arguments are defined automatically from the results. The RESULT parameter saves the result if the NAG algorithm is a function rather than a subroutine.

Some NAG algorithms may have an argument that is an external function or subroutine that performs a calculation. This can be specified for the NAG directive by supplying a pointer whose first element defines the calculation using a Genstat expression, or a pointer to several Genstat expressions. With an external function, the next element of the pointer should be the Genstat data structure that receives the result of the calculation in the expression(s). The remaining elements should be the Genstat data structures that correspond to the arguments of the external function or subroutine, in the order in which they occur in the definition of the function or subroutine in the NAG documentation. The expression or expressions are evaluated within the NAG directive by making a call to the CALCULATE directive. The ZDZ, TOLERANCE, SEED and

INDEX options of the NAG directive can be used to set the corresponding options of CALCULATE for the call.

Options: PRINT, NAME, ZDZ, TOLERANCE, SEED, INDEX.
Parameters: ARGUMENTS, RESULT.

See also
Directives: POINTER, EXPRESSION, CALCULATE, FITNONLINEAR, FLRV, QRD, SVD.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

NNDISPLAY

Displays output from a multi-layer perceptron neural network fitted by NNFIT.

Option

PRINT = *string tokens*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ

Parameter

pointers	Save structure with details of the network and the estimated parameters

Description

NNDISPLAY displays results from the fit of a neural network by NNFIT. The type of neural network fitted by NNFIT is a fully-connected feed-forward multi-layer perceptron with a single hidden layer. This network starts with a row of nodes, one for each input variable (i.e. x-variate), which are all connected to every node in the hidden layer. The nodes in the hidden layer are then all connected to the output node in the final, output layer.

Details of the fit and the structure of the neural network can be supplied using the parameter of NNDISPLAY. This must have been saved using the SAVE parameter of NNFIT. If this is not set, the output is from the most recent network fitted by NNFIT.

The output is controlled by the PRINT option, with settings:

description	a description of the network (number of input variables, nodes etc.),
estimates	estimates of the free parameters,
fittedvalues	fitted values,
summary	summary (numbers of iterations, objective function etc.).

Option: PRINT.
Parameter: unnamed.

See also

Directives: NNFIT, NNPREDICT.
Genstat Reference Manual 1 Summary section on: Data mining.

NNFIT

Fits a multi-layer perceptron neural network.

Options

PRINT = *string tokens*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ
NHIDDEN = *scalar*	Number of functions in the hidden layer; no default, must be set
HIDDENMETHOD = *string token*	Type of activation function in the hidden layer (logistic, hyperbolictangent); default logi
OUTPUTMETHOD = *string token*	Type of activation function in the output layer (linear, logistic, hyperbolictangent); default line
GAIN = *scalar*	Multiplicative constant to use in the functions; default 1
NTRIES = *scalar*	Number of times to search for a good initial starting point for the optimization; default 5
NSTARTITERATIONS = *scalar*	Number of iterations to use to find a good starting point for the optimization; default 30
VALIDATIONOPTIONS = *variate*	Variate containing three integers to control validation for early stopping; default * i.e. no early stopping; default !(10,4,16)
SEED = *scalar*	Seed for random numbers to generate initial values for the free parameters; default 0
MAXCYCLE = *scalar*	Maximum number of iterations of the conjugate-gradient algorithm; default 50

Parameters

Y = *variates*	Response variates
X = *pointers*	Input variates
YVALIDATION = *variates*	Validation data for the dependent variates
XVALIDATION = *pointers*	Validation data for the independent variates
FITTEDVALUES = *variates*	Fitted values generated for each y-variate by the neural network
OBJECTIVE = *scalars*	Value of the sum of squares objective function at the end of the optimization
NCOMPLETED = *scalars*	Number of completed iterations of the conjugate-gradient algorithm
EXIT = *scalars*	Saves the exit code
SAVE = *pointers*	Saves details of the network and the estimated parameters

Description

A neural network is a method for describing a nonlinear relationship between a response variate supplied here by the Y parameter, and a set of input variates supplied here in a pointer by the X parameter. The type of neural network fitted by NNFIT is a fully-connected feed-forward multi-layer perceptron with a single hidden layer. This network starts with a row of nodes, one for each input variable (i.e. x-variate), which are all connected to every node in the hidden layer. The nodes in the hidden layer are then all connected to the output node in the final, output layer. The number of nodes in the hidden layer is specified by the NHIDDEN option.

The output value y is given by

$$y = \psi(\Sigma_{k=1\ldots m} w_k \varphi(\Sigma_{j=1\ldots d} w_{jk} x_j - \theta) - \eta)$$

where d is the number of input nodes (i.e. x-variates),

m	is the number of hidden nodes (NHIDDEN),
x_j	is value of the jth x-variate,
w_{jk}	are weight parameters in the connections between the nodes in the input and hidden layers,
w_k	are weight parameters in the connections between the nodes in the hidden and output layer,
θ	is the threshold value subtracted at the hidden layer,
η	is the threshold value subtracted at the single node in the output layer,
$\varphi(.)$	is the activation function applied at the hidden layer,
$\psi(.)$	is the activation function applied at the output layer.

The activation functions for the hidden and outer layer are specified by the HIDDENMETHOD and OUTPUTMETHOD options, respectively, with settings:

linear	$\varphi(z) = z$ (OUTPUTMETHOD only),
logistic	$\varphi(z) = 1 / (1 + \exp(-\gamma z))$,
hyperbolictangent	$\varphi(z) = \tanh(\gamma z)$,

where the parameter γ is specified by the GAIN option; the default setting is logistic for HIDDENMETHOD, and linear for OUTPUTMETHOD.

Values for the free parameters in the multi-layer perceptron model are optimized by using a preconditioned, limited-memory quasi-Newton conjugate gradients method to minimize the objective (sum of squares) function equal to 0.5 times the average sum of squared deviation of the estimated y-values from the observed y-values.

Printed output is controlled by the PRINT option, with settings:

description	a description of the network (number of input variables, nodes etc.),
estimates	estimates of the free parameters,
fittedvalues	fitted values,
summary	summary (numbers of iterations, objective function etc.).

The NTRIES option defines the number of times to search for a good initial starting point for the optimization (default 5). The NSTARTITERATIONS option defines the number of iterations to use to find a good starting point for the optimization (default 30).

The SEED option supplies a seed for the random numbers to generate initial values for the free parameters. The default of zero continues the existing sequence of random numbers if any have already been used in the current Genstat job. If none have yet been used, Genstat picks a seed at random.

The MAXCYCLE option sets a limit on the number of iterations of the conjugate-gradient algorithm to use for the estimation (default 50).

To improve the accuracy of the neural-network approximations to new data records, it is usually desirable to stop the optimization before the value of the objective function reaches a global minimum on the training set. This method, which is known as *early stopping*, and can be performed by using a validation set of data records, specified by the YVALIDATION and XVALIDATION parameters. The optimization is then halted when the sum of squares error function achieves a minimum over the validation set of data records which has not been used to estimate the values of the free parameters in the model. The VALIDATIONOPTIONS option specifies a variate containing three integers to control validation for early stopping. The first integer defines the number of iterations of the optimizing function to complete before beginning validation; default 10. The second integer defines the number of iterations between consecutive validations; default 4. The third integer defines the number of iterations to continue validating beyond the current minimum of the objective function before stopping; default 16. This is to try to avoid the possibility of getting stuck at a local minimum. The variates in the XVALIDATION pointer must be in the same order as the corresponding variates in the X pointer.

The results of the fit, together with details about design of the neural network, can be saved using the SAVE parameter. This can then be used in the NNDISPLAY directive to display further output, or the NNPREDICT directive to form predictions.

Options: PRINT, NHIDDEN, HIDDENMETHOD, OUTPUTMETHOD, GAIN, NTRIES, NSTARTITERATIONS, VALIDATIONOPTIONS, SEED, MAXCYCLE.
Parameters: Y, X, YVALIDATION, XVALIDATION, FITTEDVALUES, OBJECTIVE, NCOMPLETED, EXIT, SAVE.

Method
NNFIT uses the function nagdmc_mlp from the Numerical Algorithms Group's library of Data Mining Components (DMCs), which estimates the free parameters using a conjugate gradient method.

Action with RESTRICT
You can restrict the set of units used for the estimation by applying a restriction to the y-variate or any of the x-variates. If several of these are restricted, they must all be restricted to the same set of units. Similarly, you can restrict the set of units used for the validation by applying a restriction to the YVALIDATION variate or any of the XVALIDATION variates.

See also
Directives: NNDISPLAY, NNPREDICT, ASRULES, RBFIT.
Procedure: KNEARESTNEIGHBOURS.
Genstat Reference Manual 1 Summary section on: Data mining.

NNPREDICT

Forms predictions from a multi-layer perceptron neural network fitted by NNFIT.

Option

PRINT = *string tokens*	Controls fitted output (description, predictions); default desc, pred

Parameters

X = *pointers*	Input variates
PREDICTIONS = *variates*	Predictions
SAVE = *pointers*	Details of the network

Description

NNPREDICT forms predictions using a neural network fitted by NNFIT. The type of neural network fitted by NNFIT is a fully-connected feed-forward multi-layer perceptron with a single hidden layer. This network starts with a row of nodes, one for each input variable (i.e. x-variate), which are all connected to every node in the hidden layer. The nodes in the hidden layer are then all connected to the output node in the final, output layer.

Details of the fit and the structure of the neural network must be supplied using the SAVE parameter. This must have been saved using the SAVE parameter of NNFIT. If this is not set, the output is from the most recent network fitted by NNFIT. The values of the input variates to be used to calculate the predictions are supplied, in a pointer, using the X parameter. The variates in the pointer must be in exactly the same order as the equivalent variates in the pointer defined for the X parameter in the original NNFIT command.

The output is controlled by the PRINT option, with settings:

description	a description of the network (number of input variables, nodes etc.),
predictions	predicted values.

Option: PRINT.
Parameters: X, PREDICTIONS, SAVE.

Method

NNPREDICT uses the function nagdmc_predict_mlp from the Numerical Algorithms Group's library of Data Mining Components (DMCs).

See also

Directives: NNDISPLAY, NNFIT.
Genstat Reference Manual 1 Summary section on: Data mining.

OPEN

Opens files.

No options

Parameters

NAME = *texts*	External names of the files
CHANNEL = *scalars*	Channel number to be used to refer to each file in other statements (numbers for each type of file are independent); if this is set to a scalar containing a missing value, the first available channel of the specified type is opened and the scalar is set to the channel number
FILETYPE = *string tokens*	Type of each file (`input, output, unformatted, backingstore, procedurelibrary, graphics`); default `inpu`
WIDTH = *scalars*	Maximum width of a record in each file; default 80
INDENTATION = *scalar*	Number of spaces to leave at the start of each line; default 0
PAGE = *scalars*	Number of lines per page (relevant only for output files)
ACCESS = *string token*	Allowed type of access (`readonly, writeonly, both`); default `both`
STYLE = *string token*	Style in which to write to an output file (`plaintext, html, latex, rtf`); default `plai`
HTMLHEAD = *texts*	Text structures containing custom content for the header of an HTML document

Description

Genstat makes use of various types of file. These are classified according to the information that they store. The files are accessed via *channels*. For each type there is a set of numbered channels that can be used to reference different files in the relevant directives. For example, there are five input channels, numbered 1 up to 5. Likewise, there are five output channels. Genstat distinguishes between the different types of channel, so you can have one file attached to output channel 3 and a different file simultaneously attached to backing store channel 3. Then, setting the option `CHANNEL=3` in `PRINT` and `STORE` statements will send the different kinds of output to the appropriate files. With backing-store files, there are six channels, numbered 0 to 5, but channel 0 is reserved for the backing-store workfile. Similarly, there are six channels, numbered 0 to 5, for unformatted files. For procedure libraries there are three channels, numbered 1 to 3. For graphics files, each channel is used for output in a particular graphics format, corresponding to the number of the device selected by the `DEVICE` directive.

When you run Genstat it starts taking input from input channel 1 and produces output on output channel 1. In an interactive run, these will be keyboard and screen, while in a batch run they will be files on the computer. Another file that is attached automatically is the start-up file of instructions that are executed at the outset of each job; this is attached to input channel 5. The start-up file may attach other files. For example, if you are working interactively, the standard start-up file arranges for output channel 5 to store a transcript of your output. (This is done using the `COPY` directive.) The command that you use to run Genstat may allow you to arrange for other files to be attached when Genstat starts running. Alternatively, within Genstat, you can use the `OPEN` directive.

Usually you need specify only the name of the file, the channel number and type of file, and leave the other parameters to take their default settings. For example, the following statements

attach a file called WEATHER.DAT to the second input channel, and then read data from it.

```
OPEN 'WEATHER.DAT'; CHANNEL=2; FILETYPE=input
READ [CHANNEL=2] Rain,Temperature,Sunshine
```

The file name can be anything that is acceptable to your computer system. You should, however, check for any constraints: for example, plotting software may require HPGL graphics files to have the extension .HPGL. You should check in your local documentation for information regarding any features that are specific to your computer or version of Genstat. For example, logical or symbolic names may be automatically translated by Genstat before files are accessed; upper and lower case characters may be significant, as on Unix systems. The file name may involve characters that have special meaning within Genstat. For example, the character \ may be required to specify directories and sub-directories on a PC. This character needs to be duplicated in a string to avoid Genstat interpreting it as the continuation symbol: for example

```
OPEN 'C:\\RES\\WEATHER.DAT'; CHANNEL=2; FILETYPE=input
```

to open the file 'C:\RES\WEATHER.DAT'. As a more convenient alternative, the PC version of Genstat allows you to use / instead.

You are free to choose which channels you want to use (within the range available for the specified type of file), apart from input and output channel 1 which are "reserved" for use by the files specified on the command line. As already mentioned, input channel 5 is used for the start-up file, and this may arrange for output channel 5 to store a transcript of your output. However, you can use the CLOSE directive to disconnect these files if you want to use the channels for some other purpose. The backing-store and unformatted work files are attached to channel 0, and this channel cannot be used in OPEN or CLOSE. Graphics files must be opened on the channel corresponding to the device number.

Obviously you cannot open more than one file on a channel, so if you wish to open a file on a channel that is currently in use you must first close that channel. Sometimes, in general programs or procedures, you may not know which channels are available. You can then let OPEN find a free channel: if CHANNEL is set to a scalar containing a missing value, the file is opened on the next available channel of the appropriate type, and the scalar is set to the number of the channel. The scalar need not be declared in advance; if CHANNEL is set to an undeclared structure, this will be defined as a scalar automatically.

```
SCALAR FreeChan
OPEN 'WEATHER.DAT'; CHANNEL=FreeChan; FILETYPE=input
READ [CHANNEL=FreeChan] Rain,Temperature,Sunshine
```

Another constraint is that you cannot open the same file on more than one channel at once.

Input files must already exist when they are opened, whereas output files will be created by Genstat. If an output file with the specified name exists already, Genstat may create an extra "version" of the file, or report a fault, or cause the file to be *overwritten*, depending on the usual conventions on your type of computer. Your local documentation will describe what rules apply in this situation, and should also explain if there are any system variables you can set to control this action.

The STYLE parameter controls the style to be used to represent the information in an output file. The default is to use plain text, which assumes that all characters occupy an equal width. So, for example, columns are aligned by use of space characters and captions are highlighted by underlining them by rows of equal signs or minuses. However, you can also choose HTML (as used for example by web browsers), RTF (as used by word processors such as Microsoft Word) or LaTeX.

When you open a file for use by backing store or unformatted input and output, you can both read from it and send output to it, unless you set the ACCESS parameter (see below). Procedure libraries are a special type of backing-store file.

The WIDTH parameter sets the maximum number of characters per line for input and output files. It is ignored for other types of file. The default values for WIDTH are designed to be

appropriate for each implementation of Genstat and may differ between input and output; details will be found in your local documentation. For input and output with screen displays that use windows WIDTH may be set automatically from the size of the appropriate window.

For input files the default is normally 80, reflecting the size of most screen displays. You can change this if necessary, to read either fewer characters from each line, or longer lines. If the WIDTH is set to be too small any extra characters will be lost, which may cause unexpected action or syntax errors. Remember that if you use READ with LAYOUT=fixed to read fixed-format data, short lines are extended with spaces up to the WIDTH setting. If you want to read data from a file with, say, 64 characters per line, setting WIDTH=64 when you open the file may make the format specification easier (rather than taking the default width of 80 and having to remember to skip 16 characters at the end of each line).

For output files, the default is the largest number of characters that can usually be displayed in a single line. This number is typically 80 for terminals but for files it is likely to be either 80, 120 or 132, depending on the type of computer. You can use the WIDTH parameter to restrict the number of output characters to a smaller number, or to a larger number up to 200.

The PAGE parameter specifies the size of page in output, affecting directives like GRAPH. For output to files, the default value of PAGE is designed to be suitable for printers. For windowed displays Genstat will, if possible, detect the size of the window and set the page size appropriately. You can also set option OUTPRINT=page in either JOB or SET to ensure that graphs and statistical analyses each start on a new page.

The INDENTATION parameter can be used to leave a specified number of blank characters to the left of each line of an output file, so that printed output can be bound for example. The indentation is subtracted from the WIDTH setting, so if you set WIDTH=80 and INDENTATION=10 then only 70 characters will be printed on each line of output.

The ACCESS parameter is used to control the way in which unformatted and backing-store files can be accessed, on computers that allow this.

The HTMLHEAD parameter allows you to supply additional markup content for the document header of an HTML file, to be inserted between the <head> and </head> tags. It can be set either to a text containing all the HTML markup or to the name of a file containing that information. It is intended primarily for inserting CSS style information, for example:

```
<style>
h1 { color: black; background-color: red !important; }
h2 { color: white; background-color: green !important; }
</style>
```

but can also be used to set any other valid header content. Additional CSS content can also be loaded via a link tag, e.g.

```
<link rel="styleSheet" type="text/css" href="genstat.css">
```

By default, the header contains a title and some standard meta data. These tags can be overwritten by specifying these tags in the inserted header
data. The tags that are treated in this way are:

```
<title>
<meta name="description"
<meta name="keywords"
<meta name="author"
```

All other content of the text is inserted verbatim and assumed to be valid HTML. If HTMLHEAD is not set, Genstat inserts the content of the file Genstat.css which is supplied with the Genstat installation in the Source directory. This defines a number of classes which are used at various points in the Genstat output (for example to define styles used for output from CAPTION). The file can be used as a template from which to derive a local variation redefining basic elements of output.

Options: none.
Parameters: NAME, CHANNEL, FILETYPE, WIDTH, INDENTATION, PAGE, ACCESS, STYLE, HTMLHEAD.

See also
Directives: CLOSE, ENQUIRE, FCOPY, FDELETE, FRENAME, READ, PRINT, INPUT, STORE, RETRIEVE, RECORD, RESUME.
Procedure: SETDEVICE.
Genstat Reference Manual 1 Summary section on: Input and output.

OPTION

Defines the options of a Genstat procedure with information to allow them to be checked when the procedure is executed.

No options

Parameters

NAME = *texts*	Names of the options
MODE = *string tokens*	Mode of each option (e, f, p, t, v, as for unnamed structures); default p
NVALUES = *scalars* or *variates*	Specifies allowed numbers of values
VALUES = *variates* or *texts*	Defines the allowed values for a structure of type variate or text
DEFAULT = *identifiers*	Default values for each option
SET = *string tokens*	Indicates whether or not each option must be set (yes, no); default no
DECLARED = *string tokens*	Indicates whether or not the setting of each option must have been declared (yes, no); default no
TYPE = *texts*	Text for each option, whose values indicate the types allowed (ASAVE, datamatrix {i.e. pointer to variates of equal lengths as required in multivariate analysis}, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, RSAVE, scalar, SSPM, symmetricmatrix, table, text, tree, TSAVE, TSM, variate, VSAVE); default * meaning no limitation
COMPATIBLE = *texts*	Defines aspects to check for compatibility with the first parameter of the directive or procedure (nvalues, nlevels, nrows, ncolumns, type, levels, labels {of factors or pointers}, mode, rows, columns, classification, margins, associatedidentifier, suffixes {of pointers}, restriction)
PRESENT = *string tokens*	Indicates whether or not each structure must have values (yes, no); default no
LIST = *string tokens*	Whether to allow a list of identifiers (MODE=p) or of values (MODE=v or t) instead of just one (yes, no); default no
INPUT = *string token*	Whether the option only supplies input information to the procedure (yes, no); default no

Description

The OPTION directive is used at the start of the definition of a Genstat procedure (initiated by the PROCEDURE directive) to define the options of the procedure. The NAMES parameter defines the names of the options. Each name also defines the identifier of a data structure that should be used, within the procedure itself, to refer to the information transmitted by the relevant option. When you use the procedure, you have the choice of typing each name in capital letters, or in small letters, or in any mixture of the two; this corresponds to the rules for the names of options and parameters of directives. Within the procedure, however, you need to be more precise, but the exact form of the identifiers will depend upon whether the Genstat environment was set to use short or long "wordlengths" when the procedure was defined. (This is controlled by the WORDLENGTH option of the JOB, SET and PROCEDURE directives.) With long wordlengths, the

identifier should be exactly the same as the option name up to the 32nd character; any characters beyond the 32nd are ignored. Alternatively, if short wordlengths have been selected, Genstat forms each identifier by truncating the corresponding option name to no more than eight characters and then converting it into capital letters.

The MODE parameter tells Genstat whether the setting of each option is to be a number (v), or an identifier of a data structure (p), or a string (t), or an expression (e), or a formula (f). These codes are exactly the same as those that indicate the mode of the values to appear within the brackets containing an unnamed structure.

The type of the structure used to represent an option of the procedure depends on the MODE and LIST parameters of the OPTION directive.

For anything other than mode p, the structure will be a dummy. This will point to an expression for mode e, a formula for mode f, and a text for mode t. With mode v, it will point to a scalar if the corresponding setting of the LIST parameter is no, and a variate if LIST=yes.

For mode p and LIST=no, the structure is a dummy, which will point to whichever structure is supplied for the option when the procedure is called; alternatively, when LIST=yes, it is a pointer which will store the list of structures that are supplied. For example, suppose that procedure ALLPOSS which contains the option definitions

```
OPTION \
  NAMES='EXP','FORM','VLN','VLY','TLN','TLY','PLN','PLY'; \
  MODE=  e,    f,     v,    v,    t,    t,    p,    p;\
  LIST=  no,   no,    no,   no,  yes,  yes,   no,  yes
```

is called with these options settings:

```
ALLPOSS [EXP=LOG10(X+1); FORM=Variety*Nitrogen; VLN=2; \
  VLY=1,3,5,7; TLN=oneval; TLY=one,two,three; \
  PLN=A; PLY=B,C,D]
```

Inside the procedure it will be as though the identifiers had been defined as follows:

```
DUMMY [VALUE=!E(LOG10(X+1))] EXP
&     [VALUE=!F(Variety*Nitrogen)] FORM
&     [VALUE=2] VLN
&     [VALUE=!(1,3,5,7)] VLY
&     [VALUE='oneval'] TLN
&     [VALUE=!T(one,two,three)] TLY
&     [VALUE=A] PLN
POINTER [VALUE=B,C,D] PLY
```

The other parameters allow the settings that are supplied, when the procedure is called, to be checked automatically.

The NVALUES parameter indicates how many values the structures that are supplied for an option of mode p may contain. For example,

```
OPTION NAME='X','Y'; NVALUES=3,!(3,4); TYPE='variate'
```

indicates that the variates supplied for X must be of length 3, while those supplied for Y can be of length 3 or 4.

The VALUES parameter can be used with modes t and v to specify an allowed set of values against which those supplied for the option will be checked. In this example, the values allowed for METHOD are Logit, Comploglogog or Angular.

```
OPTION NAME='METHOD'; MODE=t; \
  VALUES=!t(Logit,Comploglog,Angular); \
  DEFAULT='Logit'
```

The allowed values for mode t define a list of *string tokens* for the option or parameter, that can be used in exactly the same way as the string tokens defined for options or parameters of the ordinary Genstat directives. They can be up to 32 characters in length; characters 33 onwards are ignored. Each value must start with a letter, and may then contain letters or digits. When the procedure is used, Genstat will check the specified string against those in the VALUES list, using

the same abbreviation rules as for string tokens in options or parameters of the ordinary Genstat directives. Thus, for example, to request an angular transformation we need merely put `METHOD=A` as the first letter `A` is sufficient to distinguish `Angular` from `Logit` and `Comploglog`. Within the procedure, Genstat then sets `METHOD` to the full string as defined in the `VALUES` list, i.e. `Angular`, and this greatly simplifies its subsequent use. However, if short wordlengths have been requested, the name is truncated to eight characters and put into capital letters, so `Comploglog` would become `COMPLOGL`.

As an example of mode `v`, this specification would ensure that the numbers supplied for an option `NV` were all odd integers between one and nine

```
OPTION NAME='NV'; MODE=v; VALUES=!(1,3,5,7,9)
```

The `DEFAULT` parameter specifies default values to be used if the option or parameter or option is not set. Above `METHOD` will be set by default to `'Logit'`.

The `SET` parameter indicates whether or not an option must be set. The `DECLARED` parameter specifies whether or not the structures to which options of mode `p` are set must already have been declared. The `TYPE` parameter can be used to specify a text to indicate the allowed types of the structures to which an option of mode `p` is set. The `COMPATIBLE` parameter can be used to specify compatibility checks to be made for the setting of an option against the first parameter of the procedure. (The parameters are specified using the `PARAMETER` directive.) The `PRESENT` parameter allows you to indicate that the structure to which an option is set must have values. Finally, the `INPUT` parameter allows you to indicate that the option will be used only to provide input to the procedure, and will not be used to output any results. It is not essential to set this parameter but its use can improve efficiency.

For example, here the options `PERCENT` and `RESULT` can be can be either scalars, variates, tables or any type of matrix (rectangular, symmetric or diagonal). Structures to which the `PERCENT` option is set must have been declared, but for the `RESULTS` option they need not have been. Likewise the `PERCENT` option must have values, but the `RESULTS` option need not.

```
OPTION NAME='PERCENT','RESULT'; \
  MODE=p; SET=yes; DECLARED=yes,no; \
  TYPE=!t(scalar,variate,matrix,symmetric,diagonal,table); \
  PRESENT=yes,no
```

Options: none.

Parameters: `NAME`, `MODE`, `NVALUES`, `VALUES`, `DEFAULT`, `SET`, `DECLARED`, `TYPE`, `COMPATIBLE`, `PRESENT`, `LIST`, `INPUT`.

See also

Directives: `PROCEDURE`, `PARAMETER`, `CALLS`, `ENDPROCEDURE`.

Genstat Reference Manual 1 Summary section on: Program control.

OR

Introduces a set of alternative statements in a "multiple-selection" control structure.

No options or parameters

Description

A *multiple-selection* control structure consists of several alternative blocks of statements. The first of these is introduced by a CASE statement. This has a single parameter, which is an expression that must yield a single number. Subsequent blocks are each introduced by an OR statement. There can then be a final block, introduced by an ELSE statement, as in the block-if structure. The whole structure is terminated by an ENDCASE statement. Full details are given in the description of the CASE directive.

Options: none.
Parameters: none.

See also

Directives: CASE, ELSE, ENDCASE, EXIT.
Genstat Reference Manual 1 Summary section on: Program control.

OUTPUT

Defines where output is to be stored or displayed.

Options

PRINT = *string tokens*	Additions to output (dots, page, unchanged); default dots,page
DIAGNOSTIC = *string tokens*	What diagnostic printing is required (messages, warnings, faults, extra, unchanged); default faul,mess,warn
WIDTH = *scalar*	Limit on number of characters per record; default width of output file
INDENTATION = *scalar*	Number of spaces to leave at the start of each line; default 0
PAGE = *scalar*	Number of lines per page
STYLE = *string token*	Style for future output to the channel (plaintext, formatted); default * i.e. unchanged

Parameter

scalar	Channel number of output file

Description

The OUTPUT directive changes the current output channel and thus re-defines where the output will be sent by the subsequent statements in a program, until another OUTPUT statement is given (excluding any statements that use a CHANNEL option to redirect their output). Thus

```
OUTPUT 2
PRINT X
PRINT [CHANNEL=3] Y
ANOVA X
```

sends the values of X, and the analysis of X by the ANOVA statement, to the file on the second output channel, and the values of Y to the file on the third.

The PRINT option controls two aspects of the output produced for example from statistical analyses: whether a line of dots is printed at the start, and whether the output begins on a new page; this can also be controlled by the OUTPRINT option of SET. Similarly, the DIAGNOSTIC option has exactly the same effect as the DIAGNOSTIC option of SET.

The WIDTH option specifies the maximum width to be used when producing output. The default value is the width specified when the file was opened, but you can subsequently decrease it; you cannot use OUTPUT to set the width to a greater value than that specified when the file was opened. The PAGE option allows you to reset the number of lines per page.

The STYLE option is relevant if the file on the channel has been opened in a style other than plain text. (The alternatives include HTML, RTF and LaTeX; see the OPEN directive). It allows you to switch between the "formatted" style that is used by default for these files, and the ordinary plain-text representation. If the STYLE option is not specified, the style is left unchanged.

Options: PRINT, DIAGNOSTIC, WIDTH, INDENTATION, PAGE, STYLE.
Parameter: unnamed.

See also

Directives: PRINT, OPEN, COPY.
Genstat Reference Manual 1 Summary section on: Input and output.

OWN

Does work specified in Fortran subprograms linked into Genstat by the user.

Option

SELECT = *scalar*	Sets a switch, designed to allow OWN to be used for many applications; standard set-up assumes a scalar in the range 0-9; default 0

Parameters

IN = *identifiers*	Supplies input structures, which must have values, needed by the auxiliary subprograms
OUT = *identifiers*	Supplies output structures whose values or attributes are to be defined by the auxiliary subprograms

Description

To implement the OWN directive, you must get access to some of the Genstat source code. The relevant section of the code is named Module X, and is distributed with Genstat to all sites, probably in a file called X.FOR. The module consists of several Fortran 77 subprograms but to implement the OWN directive you need to modify only the subprogram called G5XZXO. This contains extensive comments that describe the way it works, and the straightforward changes that you would need to make in order to call your own subprograms. These comments are designed to be the complete documentation, and so the details are not repeated here.

The IN parameter allows you to pass values of data structures into your subprograms. Genstat will check these input structures before calling your subprograms, to ensure that they are of the right type and length for your program, and that they have been assigned values. The OUT parameter copies values calculated by your subprograms into Genstat data structures. You can arrange to define the type and length of these output structures either before or after calling your subprograms.

If the setting of the IN parameter is a list of identifiers, the OWN directive will call your subprograms more than once. Each time it will make available to your subprograms the values of one structure in the IN list, and will take information from the subprograms and put them into the corresponding structure in the OUT list. Therefore, to pass several structures at a time to your subprograms, you must put the structures into pointers. For example,

```
OWN IN=!p(A1,A2,A3),!p(B1,B2,B3); OUT=X,Y
```

will call your subprograms twice, passing information about A1, A2, A3 and X the first time, and about B1, B2, B3 and Y the second time. It does this because !p(A1,A2,A3), for example, is a single structure.

If you want to pass just one pointer to your subprograms, you must ensure that OWN does not treat the pointer as a set of structures each of which is to be passed. You can do this by constructing another pointer to hold just the identifier of the pointer that you want to pass; for example:

```
POINTER [VALUES=A,B,C] S1
OWN IN=!p(S1)
```

The SELECT option allows you to call any number of subprograms independently. Thus, you can set up OWN so that the statements

```
OWN [SELECT=1]
```

and

```
OWN [SELECT=2]
```

do totally unrelated tasks. The standard version of G5XZXO deals only with the default value, 0, of SELECT, and would need to be extended if you wanted to cater for alternative values.

However, you should be able to use much of the Fortran that deals with the default setting.

The distributed version of Genstat contains a version of the G5XZXO subprogram that carries out a simple calculation, purely for illustration of how the subroutine works. In this version, the result of

```
OWN IN=!p(V,S,M); OUT=W
```

is to shift, square and scale the values of V; that is, it does the calculation

```
W = M * (V + S)**2
```

The subprogram checks that precisely three structures are given in the pointer specified by the IN parameter, and that they are a variate and two scalars with values already present. It also checks that there is precisely one output structure, a variate; this is implicitly declared by OWN if necessary, based on the length of the input variate. Missing values in the input structures are also checked for and dealt with appropriately. The subprogram calls another one called G5XZSQ actually to carry out the transformation. To modify G5XZXO, you need to alter the details of the checks on the structures and substitute the call for one to your own subprogram.

The standard version of the G5XZXO subprogram will produce Genstat diagnostics if the checks on the input or output structures fail, or if there is not enough workspace. These diagnostics are the standard ones with codes VA, SX and SP, and are dealt with by a section at the end of the G5XZXO subprogram. You can define your own diagnostics, using the code ZZ. You are not allowed to edit the standard file of error messages that stores the one-line definitions of each diagnostic code. However, you can edit the G5XZPF subprogram which is in module X. This prints extra messages after a ZZ diagnostic; instructions for editing the subprogram are contained as comments in it.

Output from your subprograms is most easily arranged by storing the information that you want in data structures, and printing these with a PRINT statement after the OWN statement. Alternatively, you can give Fortran WRITE statements; there are standard routines in Genstat for outputting numbers and strings, but they are not described here. You should use the correct Fortran unit numbers for output, and this varies between implementations of Genstat. Note that a Fortran unit number is not the same as a Genstat channel number.

Option: SELECT.
Parameters: IN, OUT.

See also
Directives: PASS, SUSPEND.
Genstat Reference Manual 1 Summary section on: Program control.

PAGE

Moves to the top of the next page of an output file.

Option

CHANNEL = *scalar*	Channel number of file; default * i.e. current output file

No parameters

Description

PAGE arranges for future output to start on a new page. By default, PAGE works on the current output channel, but you can use the CHANNEL option if you are sending output to another file. PAGE has no effect unless output is to a file, and it achieves its effect by printing a line consisting of just the control code for a form feed (ASCII character 12). The effect of PAGE is therefore independent of the page size set by the OPEN directive.

Option: CHANNEL.
Parameters: none.

See also

Directives: SKIP, CAPTION, PRINT.
Genstat Reference Manual 1 Summary section on: Input and output.

PARAMETER

Defines the parameters of a Genstat procedure with information to allow them to be checked when the procedure is executed.

No options

Parameters

NAME = *texts*	Names of the parameters
MODE = *string tokens*	Mode of each parameter (e, f, p, t, v, as for unnamed structures); default p
NVALUES = *scalars* or *variates*	Specifies allowed numbers of values
VALUES = *variates* or *texts*	Defines the allowed values for a structure of type variate or text
DEFAULT = *identifiers*	Default values for each parameter
SET = *string tokens*	Indicates whether or not each parameter must be set (yes, no); default no
DECLARED = *string tokens*	Indicates whether or not the setting of each parameter must have been declared (yes, no); default no
TYPE = *texts*	Text for each option, whose values indicate the types allowed (ASAVE, datamatrix {i.e. pointer to variates of equal lengths as required in multivariate analysis}, diagonalmatrix, dummy, expression, factor, formula, LRV, matrix, pointer, RSAVE, scalar, SSPM, symmetricmatrix, table, text, tree, TSAVE, TSM, variate, VSAVE); default * meaning no limitation
COMPATIBLE = *texts*	Defines aspects to check for compatibility with the first parameter of the directive or procedure (nvalues, nlevels, nrows, ncolumns, type, levels, labels {of factors or pointers}, mode, rows, columns, classification, margins, associatedidentifier, suffixes {of pointers}, restriction)
PRESENT = *string tokens*	Indicates whether or not each structure must have values (yes, no); default no
INPUT = *string token*	Whether the parameter only supplies input information to the procedure (yes, no); default no

Description

The PARAMETER directive is used at the start of the definition of a Genstat procedure (initiated by the PROCEDURE directive) to define the parameters of the procedure. The NAMES parameter defines the names of the parameters. Each name also defines the identifier of a data structure that should be used, within the procedure itself, to refer to the information transmitted by the relevant parameter. When you use the procedure, you have the choice of typing each name in capital letters, or in small letters, or in any mixture of the two; this corresponds to the rules for the names of options and parameters of directives. Within the procedure, however, you need to be more precise, but the exact form of the identifiers will depend upon whether the Genstat environment was set to use short or long "wordlengths" when the procedure was defined. (This is controlled by the WORDLENGTH option of the JOB, SET and PROCEDURE directives.) With long wordlengths, the identifier should be exactly the same as the parameter name up to the 32nd character; any characters beyond the 32nd are ignored. Alternatively, if short wordlengths have been selected, Genstat forms each identifier by truncating the corresponding option name to no

more than eight characters and then converting it into capital letters. The data structures within the procedure are either all dummies or all pointers, according to the setting of the PARAMETER option of the PROCEDURE directive. If they are pointers, they store all the settings, and the procedure is called only once; if they are dummies, the procedure is called once for every item in the lists.

The other parameters of PARAMETER allow the settings that are supplied, when the procedure is called, to be checked automatically similarly to those of the OPTION directive (where more details are given). The MODE parameter tells Genstat whether the setting of each parameter is to be a number (v), or an identifier of a data structure (p), or a string (t), or an expression (e), or a formula (f). These codes are exactly the same as those that indicate the mode of the values to appear within the brackets containing an unnamed structure. The NVALUES parameter indicates how many values the structures that are supplied for a parameter of mode p may contain. The VALUES parameter can be used with modes t and v to specify an allowed set of values against which those supplied for the parameter will be checked. The DEFAULT parameter specifies default values to be used if the parameter is not set, and the SET parameter indicates whether or not a parameter must be set. The DECLARED parameter specifies whether or not the structures to which options or parameters of mode p are set must already have been declared. The TYPE parameter can be used to specify a text to indicate the allowed types of the structures to which an option or parameter of mode p is set. The PRESENT parameter allows you to indicate that the structure to which an option or parameter is set must have values. Finally, the INPUT parameter allows you to indicate that the parameter will be used only to provide input to the procedure, and will not be used to output any results. It is not essential to set INPUT but its use can improve efficiency.

Options: none.
Parameters: NAME, MODE, NVALUES, VALUES, DEFAULT, SET, DECLARED, TYPE, COMPATIBLE, PRESENT, INPUT.

See also
Directives: PROCEDURE, OPTION, CALLS, ENDPROCEDURE.
Genstat Reference Manual 1 Summary section on: Program control.

PASS

Performs tasks specified in subprograms supplied by the user, but not linked into Genstat; this directive may not be available on some computers.

Option

NAME = *text*	Filename of external executable program library and entry point function; default 'GNPASS$GNPASS'

Parameters

DATA = *pointers*	Structures whose values are to be passed to the external program, and returned
ERROR = *scalars*	Reports any errors in the external program

Description

On some computers, you can arrange that one program, such as Genstat, calls for another to be executed, with information passed directly between the two. You can thus cause Genstat to execute your own subprograms without having to modify Genstat in any way. You can do this with the PASS directive. To find out if the PASS directive has been implemented in your version, you can either look at local documentation, or issue the PASS command with no options or parameters. If it is not available, you will get a message saying that PASS has not been implemented. You could then use the SUSPEND directive instead.

To use the PASS directive, you must first compile a program library file (DLL on Windows or SO on Linux) of your own code. You will need to program the reading and writing of the data to communicate with Genstat within the functions that are called. The details are explained below.

The NAME option specifies the file name of a program library (DLL for Windows or SO for Linux) and a function name within the library, separated by a dollar symbol $. If the file extension is missing, .DLL will be appended for Windows and .SO will be appended for Linux. An advantage of omitting a file extension is that,if you have libraries that just differ in their extensions, the same command could be run on Windows and Linux. If no path is provided for the library, the system will search the Genstat Bin folder first, and then the folders in the PATH environment variable. An FI 20 fault will be given if the library is not found. The library is then searched for an exported function with the name specified after the dollar (this is case sensitive). If no function name is provided, the function GNPASS is used. An FI 21 fault is given if the function is not found.

You can use the DATA pointer to pass the values of any data structures except texts. All the structures needed by your subprograms must be combined in a pointer structure, unless only one structure is needed. The structures must have values before you include them in a PASS statement; if you want to use some of the structures to store results from your subprograms, you must initialize them to some arbitrary values, such as zero or missing. If you specify several pointers in a PASS statement, your subprograms will be invoked several times, to deal in turn with the set of structures stored by each pointer. However, the values of the structures in all the pointers are copied before any work is done by your subprograms. Thus, if you want to operate with PASS on the results of a previous operation by PASS, you must give two PASS statements with one pointer each rather than one statement with two pointers. The ERROR parameter can pass a scalar value back into Genstat to indicate whether any errors have occurred.

As an example, consider using PASS to carry out a simple transformation of a variate, as would be done by the statement

```
CALCULATE W = M*(V+S)**2
```

where V and W are variates, and M and S are scalars. Example GNPASS programs which calculate this transformation in Fortran (GENPASS.f90) and C (GNPASSC.c) are available is the Source

directory of the Genstat installation. The functions in Fortran must have arguments (INFILE, OUTFILE) of type CHARACTER(*) and in C (INFILE, OUTFILE, INLEN, OUTLEN) with the first two arguments of type char * and the last two of type int. INFILE gives the filename of the input data and OUTFILE gives the filename of the output to be read into Genstat. The extra C arguments are the lengths of the first two arguments. The example files contain a function SQUARE that calculates the transformation. The code needs to handle missing values by comparing data with the missing value indicators given in INFILE. Compiled versions of these for Windows (GNPASS.dll and GNPASSC.dll) are also in the Source folder. To create your own library you can use these as a template, replace SQUARE with your own function, and then compile and link the code into a program library. To use the Fortran example, run the following statements:

```
SCALAR S,M; VALUE=2,10
VARIATE V,W; VALUES=!(1...10),!(10(*))
TEXT Lib; VALUE='%GENDIR%/Source/GNPASS$GNPASS'
PASS [NAME=Lib] !p(V,S,M,W)
```

The PASS statement causes the program to run, and assigns the calculated values to the variate W. To use the C program you would use GNPASSC as the library.

Numbers can be used in place of scalars, as usual in Genstat statements:

```
PASS [NAME='%GENDIR%/Source/GNPASS$GNPASS'] !P(V,2,10,W)
```

To transform the values in both V, as above, and another variate X, with values 10...50 say, you could give the extra statements:

```
VARIATE X,Y; VALUES=!(10...50),!(50(*))
PASS [NAME=Lib] !p(V,2,10,W),!p(X,2,10,Y)
```

After preparing the Fortran or C program, you need to form it into an executable program, using a Fortran or C compiler. It may also be possible to use other source languages, provided the input and output formats of their compilers are compatible with that used by Genstat. All floating point values are passed from Genstat as type REAL*8 for Fortran or double for C. Factors and other items are passed as INTEGER for Fortran or int (4 bytes) for C. The GNPASS programs loop around the pointers, with the number of pointers (an integer) provided as the first value in the INFILE file. The next 3 items in the input file are the missing value representation for reals (given twice for historical reasons) and integers. Then there are sets of values for each pointer: the number of structures passed, and then the lengths of the arrays of type double precision, single precision (this is not used and should be zero) and integer. Then, for each structure in turn, the file contains its length, mode, Genstat origin and maximum block size (all integers) and its data. The Genstat origin is not used within the program, but should be written back to the result file. The maximum block size is no longer needed, as the files are not now written with a record structure but in unformatted binary mode. However, it must also be written to the result file. The Fortran program reads and writes the data in blocks, but the C program just uses single statements for this. The data are in double precision for mode 2 or integer for mode 3. The results must be written to the OUTFILE file in the same format, other than that the number of pointers is replaced by an integer error code (0 for success) which is returned in the ERROR parameter, and the missing values indicators are omitted. The example programs read the real and integer data into a single array with a calculated offset for each structure, and pass this in a common block/global structure. However, the data could be saved into individual structures and passed as arguments to the subroutine in your own program.

Option: NAME.
Parameters: DATA, ERROR.

See also
Directive: EXTERNAL, SUSPEND.

Genstat Reference Manual 1 Summary section on: Program control.

PCO

Performs principal coordinates analysis, also principal components and canonical variates analysis (but with different weighting from that used in CVA) as special cases.

Options

PRINT = *string tokens*	Printed output required (roots, scores, loadings, residuals, centroid, distances); default * i.e. no printing
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no

Parameters

DATA = *identifiers*	These can be specified either as a symmetric matrix of similarities or transformed distances or, for the canonical variates analysis, as an SSPM containing within-group sums of squares and products etc or, for principal components analysis, either as a pointer containing the variates of the data matrix or as a matrix storing the variates by columns
LRV = *LRVs*	Latent vectors (i.e. coordinates or scores), roots and trace from each analysis
CENTROID = *diagonal matrices*	Squared distances of the units from their centroid
RESIDUALS = *matrices* or *variates*	Distances of the units from the fitted space
LOADINGS = *matrices*	Principal component loadings, or canonical variate loadings
DISTANCES = *symmetric matrices*	Computed inter-unit distances calculated from the variates of a data matrix, or inter-group Mahalanobis distances calculated from a within-group SSPM
SAVE = *pointers*	Saves details of the analysis; if unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive)

Description

The PCO directive is used for principal coordinates analysis. This method encompasses principal components analysis and a form of canonical variates analysis as special cases as explained above.

There are six sections of output from PCO, requested using the PRINT option:

roots	prints the latent roots and trace;
scores	prints the principal coordinate scores;
loadings	when the directive is being used for principal components analysis or canonical variates analysis, this specifies that the loadings from the analysis are to be printed;
residuals	prints the residuals, this is relevant only if results are to be printed corresponding to only some of the latent roots;
centroid	prints the distances (not squared distances) of each unit from their overall centroid;
distances	prints the matrix of inter-unit distances (not squared distances).

The NROOTS and SMALLEST options control the printed output of roots, scores, loadings and

residuals. By default, results are printed for all the roots, but you can set the NROOTS option to specify a lesser number. If option SMALLEST has the default setting no these are taken to be the largest roots, but if you set SMALLEST=yes the results are for the smallest non-zero roots. The inter-unit distances are unaffected by the setting of the NROOTS option.

The DATA parameter supplies the data. In its simplest form, PCO works on a symmetric matrix, with values giving the associations amongst a set of objects. This could, for example, be a similarity matrix produced by FSIMILARITY.

Alternatively, the input to PCO can be a pointer whose values are the identifiers of a set of variates, or a matrix storing the variates by columns. Now the PCO directive will construct the matrix of inter-unit squared distances, and will base the analysis on associations derived from this. This is equivalent to a principal components analysis; however, the results are derived by analysing the distance matrix rather than an SSPM. When there are more units than variates, using PCO for principal components analysis is less efficient than using the PCP directive; however, if there are more variates than units the PCO directive is more efficient. When PCO is used for principal components analysis, all the variates must be of the same length and none of their values may be missing; any restrictions on the variates are ignored.

The third type of input to PCO is an SSPM structure. This must be a within-group SSPM: that is, you must have set the GROUP option of the SSPM directive when the SSPM was declared. Now the PCO directive will calculate the Mahalanobis distances amongst the group means, and base the analysis on them. This will give results similar to a canonical variates analysis. The representation of distances will be better than that of CVA, but CVA will be better if you are interested in loadings for discriminatory purposes.

The second and subsequent parameters of PCO allow you to save the results. The number of units that determine the sizes of the output structures differs according to the input to PCO. For a matrix or a symmetric matrix the number of units is the number of rows of the matrix, for a pointer it is the number of values in the variates that the pointer contains, while for an SSPM the number of units is the number of groups.

The latent roots, scores and trace can be saved in an LRV structure using the LRV parameter. If you have declared the LRV already, its number of rows must equal the number of units.

If the input to PCO is a pointer, a matrix, or an SSPM, the principal component or canonical variate loadings can be saved in a matrix using the LOADINGS parameter. The number of rows of the matrix is equal to the number of variates (either those specified by an input pointer or those specified in the SSPM directive for an input SSPM structure), or the number of columns in an input matrix.

The number of columns of the LRV and of the LOADINGS matrix corresponds to the number of dimensions to be saved from the analysis, and this must be the same for both of them. If the structures have been declared already, Genstat will take the larger of the numbers of columns declared for either, and declare (or redeclare) the other one to match. If neither has been declared and option SMALLEST retains the default setting no, Genstat takes the number of columns from the setting of the NROOTS option. Otherwise, Genstat saves results for the full set of dimensions. The trace saved as the third component of the LRV structure, however, will contain the sums of all the latent roots, whether or not they have all been saved.

The distances of the units from their centroid can be saved in a diagonal matrix using the CENTROID parameter. The diagonal matrix has the same number of rows as the number of units, defined above. The RESIDUALS parameter allows you to save residuals, formed from the dimensions that have not been saved, in a matrix with one column and number of rows equal to the number of units. Finally, the inter-unit distances can be saved in a symmetric matrix using the DISTANCES parameter. The number of rows of the symmetric matrix is again the same as the number of units.

The SAVE parameter can supply a pointer to save a multivariate save structure contining all the details of the analysis. If this is unset, an unnamed save structure is saved automatically (and

this can be accessed using the GET directive). Alternatively, you can set SAVE=* to prevent any save structure being formed if, for example, you have a very large data set and want to avoid committing the storage space.

Having obtained an ordination, you may sometimes want to add points to the ordination for additional units. If you know the squared distances of the new units from the old, the technique of Gower (1968) can be used to add points to the ordination for the new units. You can do this in Genstat by using the ADDPOINTS directive.

Options: PRINT, NROOTS, SMALLEST.
Parameters: DATA, LRV, CENTROID, RESIDUALS, LOADINGS, DISTANCES, SAVE.

Action with RESTRICT
PCO ignores any restrictions on the DATA variates.

Reference
Gower, J.C. (1968). Adding a point to vector diagrams in multivariate analysis. *Biometrika*, **55**, 582-585.

See also
Directives: CVA, FCA, MDS, PCP, PCORELATE, SSPM.
Procedures: LRVSCREE, DBIPLOT, DMST, MULTMISSING, MVAOD, DISCRIMINATE, SDISCRIMINATE, PLS, RIDGE.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

PCORELATE

Relates the observed values on a set of variates or factors to the results of a principal coordinates analysis.

Options

COORDINATES = *matrix*	Points in reduced space; no default i.e. this option must be specified
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed

Parameters

DATA = *variates* or *factors*	The data variables
TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is takenz

Description

One way of interpreting the principal coordinates obtained from a similarity matrix by PCO is by relating them to the original data variables. For each coordinate and each data variable, an F-statistic can be computed as if the variable and the coordinate vector were independent. This is not the case but, although the exact distribution of these pseudo F-values is not known, they do serve to rank the variables in order of importance of their contribution to the coordinate vector.

The DATA parameter lists the variables (variates or factors) that are to be related to the PCO results and the TEST parameter indicates their "type" as in the FSIMILARITY directive. The RANGE parameter contains a list of scalars, one for each variable in the DATA list, allowing you to standardize quantitative variates.

Qualitative variables (variates or factors with TEST settings simplematching - rogerstanimoto) are treated as grouping factors, and the mean coordinate for each group is calculated. Only 10 groups are catered for; group levels above 10 are combined. The pseudo F-statistic gives the between-group to within-group variance ratio. Missing values are excluded.

Quantitative variables (i.e. variates with other settings) are grouped on a scale of 0-10 (where zero signifies a value up to 0.05 of the range), and mean coordinates for each group are calculated. The printed pseudo F statistic is for a linear regression of the principal coordinate on the ungrouped data variate, after standardizing the data variate to have unit range; the regression coefficient is also printed.

The COORDINATES option must be present and must be a matrix. This represents the units in reduced space. Usually the coordinates will be from a principal coordinates analysis. The number of rows of the matrix must match the number of units present in the variables, taking account of any restriction.

The output from PCORELATE can be extensive. You may not be interested in relating the variables to the higher dimensions of the principal coordinates analysis even though you may have saved these in the coordinate matrix. The NROOTS option can request that results for only some of the dimensions are printed. If NROOTS is not specified, PCORELATE prints information for all the saved dimensions: that is, for the number of columns of the coordinates matrix.

(Note : this directive was originally called RELATE.)

Options: COORDINATES, NROOTS.
Parameters: DATA, TEST, RANGE.

See also
Directive: PCO.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

PCP

Performs principal components analysis.

Options

PRINT = *string tokens*	Printed output required (loadings, roots, residuals, scores, tests); default * i.e. no printing
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed
SMALLEST = *string token*	Whether to print the smallest roots instead of the largest (yes, no); default no
METHOD = *string token*	Whether to use sums of squares, correlations or variances and covariances (ssp, correlation, vcovariance, variancecovariance); default ssp

Parameters

DATA = *pointers* or *matrices* or *SSPMs*	Pointer of variates forming the data matrix, or matrix storing the variate values by columns, or SSPM giving their sums of squares and products (or correlations) etc
LRV = *LRVs*	To store the principal component loadings, roots and trace from each analysis
SSPM = *SSPMs*	To store the computed sum-of-squares-and-products or correlation matrix
SCORES = *matrices*	To store the principal component scores
RESIDUALS = *matrices* or *variates*	To store residuals from the dimensions fitted in the analysis (i.e. number of columns of the SCORES matrix, or as defined by the NROOTS option)
SAVE = *pointers*	Saves details of the analysis; if unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive)

Description

Principal components analysis finds linear combinations of a set of variates that maximize the variation contained within them, thereby displaying most of the original variability in a smaller number of dimensions. Principal components analysis operates on sums of squares and products, or a correlation matrix, or a matrix of variances and covariances, formed from the variates.

You supply the input for PCP using the first parameter; this list may have more than one entry, in which case Genstat repeats the analysis for each of the input structures. Instead of supplying an SSPM, you can supply a pointer containing the set of variates, or a matrix storing the variate values by columns. Genstat will then calculate the sums of squares and products, or correlations, or variances and covariances for the analysis (see option METHOD below).

For example, these two forms of input are equivalent:

```
SSPM [TERMS=Height,Length,Width,Weight] S
FSSPM S
PCP [PRINT=roots] S
```

and

```
PCP [PRINT=roots] !P(Height,Length,Width,Weight)
```

But the first form does mean that you have the sums of squares and products available for later use, in the SSPM S. Here the pointer is unnamed but you may wish to use a named pointer. For example:

```
POINTER [VALUES=Height,Length,Width,Weight] Dmat
PCP [PRINT=roots] Dmat
```

By default the PCP directive does not print any results: you use the PRINT option to specify what output you require. The printed output is in five sections, each with a corresponding setting, as illustrated in the examples below.

The columns of the matrices of principal component loadings and scores correspond to the latent roots. Each latent root corresponds to a single dimension, and gives the variability of the scores in that dimension. The loadings give the linear coefficients of the variables that are used to construct the scores in each dimension.

The significance tests are for equality of the k smallest roots: l_i ($i = 1, 2, \ldots k$). The test statistic is

$$n - ((2p + 11) / 6) [\log((1/k) \sum_{i>k} i_i) - (1/k) \sum_{i>k} \log(i_i)]$$

where n is the number of units and p is the number of variables. Asymptotically, the statistics have a chi-square distribution with $(k+2)(k-1)/2$ degrees of freedom. If any latent roots are zero, Genstat excludes them from the calculation of the test statistic; the effective value of p is reduced accordingly.

If you omit the NROOTS option, Genstat prints by default the results corresponding to all the latent roots. The number of latent roots is the number of variates involved in the input to PCP. The NROOTS option allows you to print only part of the results, corresponding to the first or last r latent roots. You may then want to print the residuals formed from the remaining columns of scores. The residuals are all positive: this is because residuals from multivariate analyses generally occupy several dimensions, so they represent distances in multidimensional space and signs cannot be attached to them.

To print results corresponding to the r smallest latent roots, you must set option NROOTS to r and option SMALLEST to yes. Now if residuals are printed they will be formed from the scores corresponding to the largest roots. The NROOTS and SMALLEST options apply to the latent roots and vectors, the principal component scores and the residuals. So you cannot print directly, for example, the first two columns of scores and the last three columns of loadings. This is rarely required but, if necessary, it can be done by saving the relevant results and printing them separately.

By default, the PCP directive operates on the SSPM but you can set the METHOD option to correlations to operate on a derived matrix of correlations, or to vcovariance (or its synonym variancecovariance) to use variances and covariances. Note that when correlations are analysed the significance-test statistics no longer have asymptotic chi-square distributions.

The LRV parameter allows you to save the principal component loadings, the latent roots, and their sum (the trace) in an LRV structure, while the SCORES parameter saves the principal component scores in a matrix. If you have declared the LRV already, its number of rows must be the same as the number of variates supplied in an input pointer or implied by an input SSPM. The number of rows of the SCORES matrix, if previously declared, must be equal to the number of units.

The number of columns of the LRV and of the SCORES matrix corresponds to the number of dimensions to be saved from the analysis, and this must be the same for both of them. If the structures have been declared already, Genstat will take the larger of the numbers of columns declared for either, and declare (or redeclare) the other one to match. If neither has been declared and option SMALLEST retains the default setting no, Genstat takes the number of columns from the setting of the NROOTS option. Otherwise, Genstat saves results for the full set of dimensions. The trace saved as the third component of the LRV structure, however, will contain the sums of all the latent roots, whether or not they have all been saved. Procedure LRVSCREE can be used to produce a "scree" diagram which can be helpful in deciding how many dimensions to save.

The SSPM parameter can save the SSPM structure used for the analysis. A particularly

convenient instance is when you have supplied an SSPM structure as input but, for example, have set METHOD=correlation: the SSPM that is saved will then contain correlations instead of sums of squares and products.

The RESIDUALS parameter allows you to save the principal component residuals, in a matrix with number of rows equal to the number of units and one column. If the latent roots and vectors (loadings) are saved from the analysis, the residuals will correspond to the dimensions not saved; the same applies if you save scores. If neither the LRV nor scores are saved, the saved residuals will correspond to the smallest latent roots not printed.

The SAVE parameter can supply a pointer to save a multivariate save structure contining all the details of the analysis. If this is unset, an unnamed save structure is saved automatically (and this can be accessed using the GET directive). Alternatively, you can set SAVE=* to prevent any save structure being formed if, for example, you have a very large data set and want to avoid committing the storage space.

If you want principal component scores or residuals to be printed or saved from the analysis, the original data must be available. The matrices to save such results must have been declared with as many rows as the variates have values, ignoring the restriction. You can calculate the analysis from one subset of units, but calculate the scores and residuals for all the units, by using as input to PCP an SSPM structure formed using a weight variate with zeros for the excluded sampling units and unity for those to be included. For example, to exclude a known set of outliers from an analysis, but to print scores for them, these statements could be used:

```
POINTER [NVALUES=5] V
FACTOR [LABELS=!T(No,Yes)] Outlier
READ [CHANNEL=2] Outlier,V[]
CALCULATE Wt = Outlier .IN. 'No'
SSPM [TERMS=V] S
FSSPM [WEIGHT=Wt] S
PCP [PRINT=scores] S
```

Principal component regression is provided by procedure RIDGE.

Options: PRINT, NROOTS, SMALLEST, METHOD.
Parameters: DATA, LRV, SSPM, SCORES, RESIDUALS, SAVE.

Action with RESTRICT

If the variables used to form the SSPM structure are restricted, then the analysis will be subject to that restriction. Similarly, if a pointer to a set of variates is used as input to PCP, then any restriction on the variates will be taken into account by the analysis.

See also

Directives: CVA, FCA, MDS, PCO, ROTATE, SSPM.
Procedures: LRVSCREE, DBIPLOT, DMSTPLS, RIDGE.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

PEN

Defines the properties of "pens" for high-resolution graphics.

Options

`RESET` = *string token*	Whether to reset the pen definitions to their default values (`yes`, `no`); default `no`
`BOXUNITS` = *string token*	Units to use for text boxes (`characters`, `distance`); the default is to retain the existing setting

Parameters

`NUMBER` = *scalars*	Numbers associated with the pens
`COLOUR` = *texts* or *scalars*	Colour to use with each pen unless otherwise specified by the `CSYMBOL`, `CLINE`, `CFILL` or `CAREA` parameters
`LINESTYLE` = *texts or scalars*	Style for line used by each pen when joining points
`METHOD` = *string tokens*	Method for determining line (`point`, `line`, `monotonic`, `closed`, `open`, `fill`, `spline`, `polygon`)
`SYMBOL` = *texts*, *scalars*, *pointers* or *matrices*	Defines the plotting symbol for each pen, by a text or scalar for a pre-defined symbol, a pointer for a user-defined symbol, or a matrix to supply a bitmap
`LABELS` = *texts* or *factors*	Define labels that will be printed alongside the plotting symbols
`ROTATION` = *scalars* or *variates*	Rotation required for the plotting symbols and labels (in degrees)
`JOIN` = *string tokens*	Order in which points are to be joined by each pen (`ascending, given`)
`BRUSH` = *scalars*	Number of the type of area filling used with each pen when drawing pie charts or histograms (unavailable in Genstat *for Windows*)
`FONT` = *texts* or *scalars*	Font to be used for any text written by each pen
`THICKNESS` = *scalars*	Thickness with which any lines are drawn by each pen
`SIZEMULTIPLIER` = *scalars* or *variates*	Multiplier used in the calculation of the size in which to draw symbols and labels by each pen unless otherwise specified by `SMSYMBOL` or `SMLABEL`
`CSYMBOL` = *texts* or *scalars*	Colour to use with each pen when drawing symbols
`CLINE` = *texts* or *scalars*	Colour to use with each pen when drawing lines
`CFILL` = *texts* or *scalars*	Colour to use with each pen when filling areas inside hollow symbols
`CAREA` = *texts* or *scalars*	Colour to use with each pen when filling areas inside polygons and bars of histograms
`SMSYMBOL` = *scalars* or *variates*	Multiplier used in the calculation of the size in which to draw symbols by each pen
`SMLABEL` = *scalars* or *variates*	Multiplier used in the calculation of the size in which to draw labels by each pen
`DFSPLINE` = *scalars*	Number of degrees of freedom to use when `METHOD=spline`
`YMISSING` = *string token*	How to treat missing y-values when `METHOD=spline` (`break`, `interpolate`)
`XMISSING` = *string token*	How to treat missing x-values when `METHOD=spline` (`break`, `ignore`)

YLPOSITION = *string token*	How to position labels in the y-direction with respect to the points (above, centre, below, automatic)
XLPOSITION = *string token*	How to position labels in the x-direction with respect to the points (left, centre, right, automatic)
YLSIZE = *scalars* or *variates*	Sizes of the y-direction of the text boxes into which to plot labels
XLSIZE = *scalars* or *variates*	Sizes of the x-direction of the text boxes
YLOFFSET = *scalars* or *variates*	Offsets in the y-direction of the text boxes
XLOFFSET = *scalars* or *variates*	Offsets in the x-direction of the text boxes
BARTHICKNESS = *scalars*	Thickness with which any error bars are drawn by each pen
BARCAPWIDTH = *scalars*	Width of the cap drawn by each pen at the top and bottom of any error bars
DESCRIPTION = *texts*	Description for points plotted by the pen, to be used by the Data Information tool in the Graphics Viewer
TSYMBOL = *scalars*	Defines the transparency of symbols drawn by each pen, on a scale of 0 (opaque) to 255 (completely transparent)
TLINE = *scalars*	Defines the transparency of lines drawn by each pen
TFILL = *scalars*	Defines the transparency to use when filling areas inside hollow symbols with each pen
TAREA = *scalars*	Defines the transparency to use when filling areas inside polygons and bars of histograms with each pen
SAVE = *pointers*	Saves details of the current settings for the pen concerned

Description

Graphical displays are drawn using graphical *pens*. The graphics commands each have particular pens that they use by default, but most allow you to specify alternatives (see DGRAPH, D3GRAPH, DCONTOUR, DHISTOGRAM etc). The attributes of the pens, such as colour, font and symbol-type, determine how the plots are generated. The PEN directive allows you to change these attributes, so that you can modify the resulting display. Different attributes are relevant for different types of plot: for example symbols and labels are used only within DGRAPH and D3GRAPH (and the graphics procedures that use them to construct their plots).

The NUMBER parameter lists the numbers of the pens, in the range 1 to 256 or −1 to −12, that you wish to redefine. By default, any aspects of these pens that are not set explicitly retain the values that they had immediately before the PEN statement. Alternatively, you can specify option RESET=yes to reset their definitions to the default values defined by Genstat at the start of each job.

Pens 1 to 256 are used for the information that is plotted in a graph (points, lines, and so on). In most of the graphics commands, the default is to use these pens in succession for the different structures that are plotted, so that the various data sets can easily be distinguished. The negatively numbered pens are used as the initial defaults for the axes and their associated marks and labels (see XAXIS), or for gridlines, overall title and key (see FRAME), or for default gridlines in shade plots (see DSHADE), or for default outlines in histograms, bar charts and pie charts (see DHISTOGRAM, BARCHART and DPIE), or for error bars (see BARCHART), or for the overall title (see DSTART). Details are given in the *Methods* Section below. They cannot be used for any other purposes.

The COLOUR, CSYMBOL, CLINE, CFILL and CAREA parameters specify the colours to be used by the pen. The COLOUR parameter defines the colour for everything plotted by the pen apart from the colour for filling "hollow" symbols (e.g. a circle but not a cross), while the other parameters define specific aspects (overriding any setting of COLOUR): CSYMBOL defines the

colour to be used for drawing symbols, CLINE defines the colour for lines, CFILL defines the colour for filling areas inside "hollow" symbols, and CAREA defines the colour for filling areas inside polygons and bars of histograms. The parameters can be set any of the following: a text containing the name of one of Genstat's pre-defined colours; a scalar containing a number defining a colour using the RGB system; or a hexadecimal digit defined in a string of the form '#rgb', '0xrgb' or '0Xrgb' where rgb are the pairs of hexadecimal digits 00-FF that give the red, green and blue intensities of the colour respectively. For example, '#FF0000', '#00FF00' and '#0000FF' give pure red, green and blue respectively. The leading zeros can be dropped, so '#FF00' and '#FF' also define green and blue respectively. You can use the RGB function to construct these colour numbers from their red, green and blue components: for example

```
CALCULATE xgold = RGB(255; 215; 0)
PEN 2; CSYMBOL=xgold
```

sets xgold to the colour gold (which has red, green and blue values 255, 215 and 0 respectively), and uses this as the colour for symbols drawn by pen 2. The numbers give you access to the complete spectrum supported by most colour graphics devices. (Note, though, that the colours are mapped automatically onto a grey scale if the device is defined with a grey-scale palette; see DEVICE). Alternatively, the pre-defined colours define the standard colours used by many web browsers, and mainly use the same names. The names, and their corresponding red, green and blue values, are listed in *Methods* section. They can be given in either upper or lower case, or in any mixture, but they must not be abbreviated.

There are two special strings that can be used for colours. The string 'background' uses the colour defined in the BACKGROUND option or parameter of FRAME. The string 'match' which can be used with CFILL to take the colour from CSYMBOL, or with CAREA to take the colour from CLINE. For example,

```
PEN 1,2,3; COLOUR='red','blue','green'; CFILL='match'
PEN 4,5,6; CLINE='red','blue','green'; CAREA='match'
```

plots filled symbols in the same colour as their outlines for pens 1 to 3, and filled areas in the same colour as their outlines for pens 4 to 5. Note, COLOUR sets all of CSYMBOL, CLINE and CAREA to the same value, so you only need to set CFILL='match' to set all colours of a pen to the same value. Also, if you want all your symbols filled, you could specify

```
PEN 1...256; CFILL='match'
```

You can also use the number −1 to specify the background colour. A missing value represents a hollow symbol for CFILL or the background colour for CSYMBOL, CLINE and CAREA.

The TSYMBOL, TLINE, TFILL and TAREA parameters accompany the parameters CSYMBOL, CLINE, CFILL and CAREA, respectively, and define the transparency of the corresponding colours. Their values can range from 0 (opaque) to 255 (completely transparent). The pens all have initial defaults of 0 for the transparencies.

The SYMBOL parameter defines the symbol that is drawn at each point, for example by DGRAPH. You can mark different points with different symbols (for example to indicate groupings in the data) by setting the PEN parameter of DGRAPH to a variate or factor specifying a pen with the appropriate symbol for each point.

Genstat provides a choice of standard symbols, listed below, that can be specified either by giving the name (in a text with a single value), or the number (in a scalar).

1 'Cross'
2 'Circle'
3 'Plus'
4 'Star'
5 'Square'
6 'Diamond'

```
 7  'Triangle'
 8  'Nabla'
 9  'Asterisk'
 10 'Minus'
 11 'Heavyminus'
 12 'Heavyplus'
 13 'Heavycross'
 14 'Smallcircle'
 15 'Tinycircle'
 16 'Female'
 17 'Male'
 18 'Rhombus'
 19 'Circlecross'
 20 'Circleplus'
 21 'Squarecross'
 22 'Squareplus'
 -1 'Sphere'
 -2 'Cone'
 -3 'Cylinder'
 -4 'Cube'
```

The final four symbols (numbered −1 to −4) are intended mainly for 3-dimensional plots, and may not be available on some devices. You can set `SYMBOL='none'` or `SYMBOL=0` if you do not want to plot symbols at the data points, as for example if you only want to draw a line through the points. You can also use `SYMBOL=0` together with the `LABELS` parameter (described below) to plot a character at the data points instead of a symbol. For example

```
PEN 1; SYMBOL='none'; LABEL='A'
```

or

```
PEN 1; SYMBOL=0; LABEL='A'
```

will identify the points plotted by pen 1 with the letter `A`.

To define a symbol of your own, you can set `SYMBOL` to a pointer containing a pair of variates defining the coordinates of a set of points to be joined by straight line segments. The points should be within a notional square with bounds −1.0 to 1.0 in each direction. The square is centred on the data point, and scaled to the same size as the standard symbols. Missing values can be included in the coordinates, to use separate pen strokes to draw the line segments. The final possibility is to set `SYMBOL` to a matrix of RGB colour values, representing a bitmap.

The `LABELS` parameter allows you to label each point with a string or number. You can set it to a text structure to specify strings to be plotted at each point. If the text has a single string, this will be be plotted at every point; otherwise the text must have the same number of values as the Y and X variates that are to be plotted. You can also set `LABELS` to a factor. If the factor has labels, then these are used; otherwise the points are labelled by its levels. This provides another way of representing grouped data.

The positioning of the labels with respect to the points is controlled by the `YLPOSITION` and `XLPOSITION` parameters. The initial default is to determine the positions automatically according to their type (e.g. labels for points, or for tick marks on the y-axis, or on the x-axis, and so on). The labels are plotted into text boxes whose widths in the x- and y-directions can be defined by the `YLSIZE` and `XLSIZE` parameters; if these are not set, the boxes are defined to plot the labels as a single line of characters. The amounts by which the boxes are offset in the the x- and y-directions can be defined by the `YLOFFSET` and `XLOFFSET` parameters; if these are not set, the positions of the boxes are defined automatically as appropriate for the positions defined by the `YLPOSITION` and `XLPOSITION` parameters. The `BOXUNITS` option defines what units to use

when defining the sizes and positions of the text boxes. The initial default is to measure these in numbers of characters of an average width in the defined font and size (see parameters FONT and SMLABEL), but you can set option BOXUNITS=distance to use the distances as defined by the axes of the graph.

The graphical symbols are drawn so that they are centred at the specified position. If LABELS are specified they are aligned alongside the markers, unless you have set SYMBOL=0 to suppress the markers, in which case the labels start from the specified (x,y) position. For compatibility with previous releases of Genstat you can also set SYMBOL to a factor, which has the same effect as setting LABELS with SYMBOL=0.

The Genstat Graphics Viewer with Genstat *for Windows* has a "Data Information" tool that allows you to display information about each point when you place the cursor over the point. If you want to replace the default information, you can set the DESCRIPTION parameter to a text (with one line for each point) containing your own information.

The METHOD parameter specifies the type of object to be plotted: points, lines or filled polygons. The initial default for every pen, METHOD=point, will result in points being plotted using the corresponding symbols, labels, colours and fonts. Various types of line can be drawn through the plotted points; either straight lines (line and polygon) or smooth curves (monotonic, open, closed and spline). The line and polygon settings differ in that, with polygon, a line is drawn also to connect the first and last points. The monotonic setting specifies that a smooth single-valued curve is to be drawn through the data points. The name is derived from the requirement that the x-values (rather than the fitted curve) must be strictly monotonic, so that there is only one y-value for each distinct x-value. To ensure this, a copy of the data is made and sorted before the curve is fitted. This setting is recommended for plotting curves fitted to data, for example with FITCURVE. You should ensure that the points are close enough for the plotted line to be a reasonable approximation. When you know the functional form of the curve, it may be advantageous to calculate extra points. The open and closed settings specify that a smooth, possibly multi-valued, curve is to be drawn through the data points, using the method of McConalogue (1970); the resulting curve is rotationally invariant, although it is not invariant under scaling. The closed setting connects the last point to the first. McConalogue's method (open or closed) is more suited to the situation where the plotted curve is intended to represent the shape of an object. Alternatively, the spline setting plots a smoothing spline fitted through the points. The DFSPLINE parameter specifies how many degrees of freedom to use in the spline (initial default 4). The YMISSING parameter controls whether to break the spline at a missing y-value or to interpolate y-value, and the XMISSING parameter controls whether to break the spline at a missing x-value or to ignore the point; the initial default for both parameters is to break the spline. The setting METHOD=fill joins the data points by straight lines to produce one or more polygons. Each polygon is then filled in the colour specified by CFILL (see below). The plotting method also determines how contours will be drawn. Also, the combination of SYMBOL=0 and METHOD=point will produce no plotting at all (and no warning) within DGRAPH.

If the requested plotting method produces a line through the points, the LINESTYLE parameter will specify what sort of line is drawn (for example a solid, dotted or dashed line). The type of style can be specified either by giving the name (in a text with a single value), or the number (in a scalar).

1 'Solid'
2 'Dot' or '.'
3 'Dash' or '-'
4 'Dotdash' or '.-'
5 'Tightdash' or 'T-'
6 'Longdash' or 'L-'
7 'Shortdash' or 'S-'

8 'Closedot' or 'C-'
9 'Finedot' or 'F.'
10 'Doubledotdash' or '..-'

Each text can all be abbreviated to the minimum number of characters required to distinguish it from the earlier styles.

The JOIN parameter controls the order in which points are connected when lines are to be drawn or the points define a polygon to be shaded. Given requests that the data are to be plotted in the order in which they are stored, whereas ascending implies that the data are copied and sorted so that the x-values are in ascending order before plotting. This parameter is ignored when METHOD=monotonic, as this requires that the data must always be sorted.

On some devices the BRUSH parameter allowed you to control how areas are shaded when METHOD is set to fill, or when plotting histograms and pie charts. There were 16 available patterns indicated by the integers 1 to 16. In general, the higher the number, the denser the hatching. In Genstat *for Windows* BRUSH is unavailable, and the areas are shaded in full. The CFILL parameter defines which colour is used by the pen to fill the areas.

The THICKNESS parameter allows you to specify an amount by which the standard thickness of plotted lines is to be multiplied. This allows you to increase the thickness of lines, perhaps to highlight some feature of a plot. You can also use thickness to emphasize the axes, by redefining the appropriate pen. For some devices, it is not possible to control the thickness of plotted lines; the THICKNESS parameter is then ignored. Similarly, the BARTHICKNESS parameter can provide a multiplier for the line thickness when the pen is used to draw an error bar, and the BARCAPWIDTH parameter can give a multiplier to adjust the standard width of the lines at the top and bottom of error bars.

The default sizes of symbols and the characters in labels are determined from the dimensions of the current window. The SIZEMULTIPLIER parameter can be used to modify the sizes of both of these, by specifying a value by which this default size is to be multiplied. Alternatively, you can use the SMSYMBOL parameter to modify just the symbol size, or the SMLABEL parameter to modify just the size of characters in labels. For example when plotting a graph in a small window you may wish to increase the size of annotation in order to make it legible. They can each be set to a scalar, or to a variate to allow the different points to be scaled in different ways.

The ROTATION parameter controls the angle (in degrees) at which to plot text or user-defined symbols. The initial setting of zero will produce text "conventionally" orientated. You can set ROTATION to a scalar value that will apply to all points, or to a variate that allows a different angle to be used at each point.

The FONT parameter defines the font family to be used by each pen to plot textual information, for example, in titles, axis annotation, plotting symbols and keys. This can be set to a text containing the name of a font family, or to a scalar containing an integer between 1 and 25. The initial default for each pen is to use the *default graphics font*, which can be defined either by using menus in the Genstat Client or Graphics Viewer, or by using the DFONT directive. You can find out the names of the fonts, available to specify in a text, by looking at any of the controls for specifying fonts in the Client or Graphics Viewer. The integers refer to fonts that should always be available. You can list these using the DHELP procedure. Font 1 has a special status, in that it automatically maps to the currently-defined default graphics font. If you change the default graphics font, this will be used as the default font in any graphs that are then displayed or redisplayed, including those that have been stored in Genstat graphics meta files (i.e. files with the gmf suffix). If you specify a font that is unavailable on your computer, the default font is used instead.

The current settings of each pen can be saved in a pointer supplied by the SAVE parameter. The elements of the pointer are labelled to identify the components. Initial default settings are represented by missing values; the actual values used for these attributes when plotting will depend on the output device.

Options: RESET, BOXUNITS.

Parameters: NUMBER, COLOUR, LINESTYLE, METHOD, SYMBOL, LABELS, ROTATION, JOIN, BRUSH, FONT, THICKNESS, SIZEMULTIPLIER, CSYMBOL, CLINE, CFILL, CAREA, SMSYMBOL, SMLABEL, DFSPLINE, YMISSING, XMISSING, YLPOSITION, XLPOSITION, YLSIZE, XLSIZE, YLOFFSET, XLOFFSET, BARTHICKNESS, BARCAPWIDTH, DESCRIPTION, TSYMBOL, TLINE, TFILL, TAREA, SAVE.

Method

The names of the standard pre-defined colours are listed below with their corresponding red, green and blue values for use e.g. in the RGB function.

Red colors

```
IndianRed                RGB(205;  92;  92)
LightCoral               RGB(240; 128; 128)
Salmon                   RGB(250; 128; 114)
DarkSalmon               RGB(233; 150; 122)
LightSalmon              RGB(255; 160; 122)
Crimson                  RGB(220;  20;  60)
Red                      RGB(255;   0;   0)
FireBrick                RGB(178;  34;  34)
DarkRed                  RGB(139;   0;   0)
```

Pink colors

```
Pink                     RGB(255; 192; 203)
LightPink                RGB(255; 182; 193)
HotPink                  RGB(255; 105; 180)
DeepPink                 RGB(255;  20; 147)
MediumVioletRed          RGB(199;  21; 133)
PaleVioletRed            RGB(219; 112; 147)
```

Orange colors

```
LightSalmon              RGB(255; 160; 122)
Coral                    RGB(255; 127;  80)
Tomato                   RGB(255;  99;  71)
OrangeRed                RGB(255;  69;   0)
DarkOrange               RGB(255; 140;   0)
Orange                   RGB(255; 165;   0)
```

Yellow colors

```
Gold                     RGB(255; 215;   0)
Yellow                   RGB(255; 255;   0)
LightYellow              RGB(255; 255; 224)
LemonChiffon             RGB(255; 250; 205)
LightGoldenrodYellow     RGB(250; 250; 210)
PapayaWhip               RGB(255; 239; 213)
Moccasin                 RGB(255; 228; 181)
PeachPuff                RGB(255; 218; 185)
PaleGoldenrod            RGB(238; 232; 170)
Khaki                    RGB(240; 230; 140)
DarkKhaki                RGB(189; 183; 107)
```

Purple colors

```
Lavender                 RGB(230; 230; 250)
Thistle                  RGB(216; 191; 216)
Plum                     RGB(221; 160; 221)
Violet                   RGB(238; 130; 238)
Orchid                   RGB(218; 112; 214)
Fuchsia                  RGB(255;   0; 255)
Magenta                  RGB(255;   0; 255)
MediumOrchid             RGB(186;  85; 211)
MediumPurple             RGB(147; 112; 219)
BlueViolet               RGB(138;  43; 226)
```

DarkViolet	RGB(148; 0; 211)
DarkOrchid	RGB(153; 50; 204)
DarkMagenta	RGB(139; 0; 139)
Purple	RGB(128; 0; 128)
Indigo	RGB(75; 0; 130)
SlateBlue	RGB(106; 90; 205)
DarkSlateBlue	RGB(72; 61; 139)

Green colors

GreenYellow	RGB(173; 255; 47)
Chartreuse	RGB(127; 255; 0)
LawnGreen	RGB(124; 252; 0)
Lime	RGB(0; 255; 0)
LimeGreen	RGB(50; 205; 50)
PaleGreen	RGB(152; 251; 152)
LightGreen	RGB(144; 238; 144)
MediumSpringGreen	RGB(0; 250; 154)
SpringGreen	RGB(0; 255; 127)
MediumSeaGreen	RGB(60; 179; 113)
SeaGreen	RGB(46; 139; 87)
ForestGreen	RGB(34; 139; 34)
Green	RGB(0; 128; 0)
DarkGreen	RGB(0; 100; 0)
YellowGreen	RGB(154; 205; 50)
OliveDrab	RGB(107; 142; 35)
Olive	RGB(128; 128; 0)
DarkOliveGreen	RGB(85; 107; 47)
MediumAquamarine	RGB(102; 205; 170)
DarkSeaGreen	RGB(143; 188; 143)
LightSeaGreen	RGB(32; 178; 170)
DarkCyan	RGB(0; 139; 139)
Teal	RGB(0; 128; 128)

Blue colors

Aqua	RGB(0; 255; 255)
Cyan	RGB(0; 255; 255)
LightCyan	RGB(224; 255; 255)
PaleTurquoise	RGB(175; 238; 238)
Aquamarine	RGB(127; 255; 212)
Turquoise	RGB(64; 224; 208)
MediumTurquoise	RGB(72; 209; 204)
DarkTurquoise	RGB(0; 206; 209)
CadetBlue	RGB(95; 158; 160)
SteelBlue	RGB(70; 130; 180)
LightSteelBlue	RGB(176; 196; 222)
PowderBlue	RGB(176; 224; 230)
LightBlue	RGB(173; 216; 230)
PurwaBlue	RGB(155; 225; 255)
SkyBlue	RGB(135; 206; 235)
LightSkyBlue	RGB(135; 206; 250)
DeepSkyBlue	RGB(0; 191; 255)
DodgerBlue	RGB(30; 144; 255)
CornflowerBlue	RGB(100; 149; 237)
MediumSlateBlue	RGB(123; 104; 238)
RoyalBlue	RGB(65; 105; 225)
Blue	RGB(0; 0; 255)
MediumBlue	RGB(0; 0; 205)
DarkBlue	RGB(0; 0; 139)
Navy	RGB(0; 0; 128)
MidnightBlue	RGB(25; 25; 112)

Brown colors

Cornsilk	RGB(255; 248; 220)

BlanchedAlmond	RGB(255; 235; 205)
Bisque	RGB(255; 228; 196)
NavajoWhite	RGB(255; 222; 173)
Wheat	RGB(245; 222; 179)
BurlyWood	RGB(222; 184; 135)
Tan	RGB(210; 180; 140)
RosyBrown	RGB(188; 143; 143)
SandyBrown	RGB(244; 164; 96)
Goldenrod	RGB(218; 165; 32)
DarkGoldenrod	RGB(184; 134; 11)
Peru	RGB(205; 133; 63)
Chocolate	RGB(210; 105; 30)
SaddleBrown	RGB(139; 69; 19)
Sienna	RGB(160; 82; 45)
Brown	RGB(165; 42; 42)
Maroon	RGB(128; 0; 0)

White colors

White	RGB(255; 255; 255)
Snow	RGB(255; 250; 250)
Honeydew	RGB(240; 255; 240)
MintCream	RGB(245; 255; 250)
Azure	RGB(240; 255; 255)
AliceBlue	RGB(240; 248; 255)
GhostWhite	RGB(248; 248; 255)
WhiteSmoke	RGB(245; 245; 245)
Seashell	RGB(255; 245; 238)
Beige	RGB(245; 245; 220)
OldLace	RGB(253; 245; 230)
FloralWhite	RGB(255; 250; 240)
Ivory	RGB(255; 255; 240)
AntiqueWhite	RGB(250; 235; 215)
Linen	RGB(250; 240; 230)
LavenderBlush	RGB(255; 240; 245)
MistyRose	RGB(255; 228; 225)

Grey colors

Gainsboro	RGB(220; 220; 220)
LightGray or LightGrey	RGB(211; 211; 211)
Silver	RGB(192; 192; 192)
DarkGray or DarkGrey	RGB(169; 169; 169)
Gray or Grey	RGB(128; 128; 128)
DimGray or DimGrey	RGB(105; 105; 105)
LightSlateGray or LightSlateGrey	RGB(119; 136; 153)
SlateGray or SlateGrey	RGB(112; 128; 144)
DarkSlateGray or DarkSlateGrey	RGB(47; 79; 79)
Black	RGB(0; 0; 0)

In addition, the string `Background` can be used to refer to the background colour that has been defined (e.g. by `FRAME`) for the particular part of the screen where the pen is being used, and the string `Transparent` can be used for example to define a colour that will not obscure anything that is plotted below it (e.g. in another window). Alternatively, you can define the colour intensity with three pairs of hexadecimal digits (00-FF), by using a string that starts with either `#`, `0x` or `0X`, and then contains the three pairs of digits: i.e. you can specify either `'#rgb'`, `'0xrgb'` or `'0Xrgb'`, where `rgb` are the pairs of hexadecimal digits that give the red, green and blue intensities of the colour respectively.

The roles of the negatively-numbered pens are listed below:

−1 initial default for `PENTITLE` parameter of `XAXIS`, `YAXIS` and `ZAXIS`

-2 initial default for PENAXIS parameter of XAXIS, YAXIS and ZAXIS
-3 initial default for PENLABEL parameter of XAXIS, YAXIS and ZAXIS
-4 initial default for PENGRID parameter of FRAME
-5 initial default for PENTITLE parameter of FRAME
-6 initial default for PENKEY parameter of FRAME
-7 default for PENGRID option of DSHADE
default for PENGRID option of DBITMAP
-8 default for PENOUTLINE option of DHISTOGRAM
-9 default for PENOUTLINE option of BARCHART
-10 default for PENOUTLINE option of DPIE
-11 default for XBARPEN and YBARPEN parameters of DGRAPH
default for PENERRORBARS parameter of BARCHART
-12 pen for title in DSTART

Reference

McConalogue, D.J. (1970). A quasi-intrinsic scheme for passing a smooth curve through a discrete set of points. *Computer Journal*, **13**, 392-396.

See also

Directives: DFONT, DLOAD, DSAVE, COLOUR.
Procedures: DHELP, GETRGB.
Functions: BLUE, GREEN, GRAY, GREY, RED, RGB.
Genstat Reference Manual 1 Summary section on: Graphics.

POINTER

Declares one or more pointer data structures.

Options

`NVALUES` = *scalar* or *text*	Number of values, or labels for values; default `*`
`VALUES` = *identifiers*	Values for all the pointers; default `*`
`SUFFIXES` = *variate* or *scalar*	Defines an integer number for each of the suffixes; default `*` indicates that the numbers 1,2,... are to be used
`CASE` = *string token*	Whether to distinguish upper and lower case in the labels of the pointers (`significant`, `ignored`); default `sign`
`ABBREVIATE` = *string token*	Whether or not to allow the labels to be abbreviated (`yes`, `no`); default `no`
`FIXNVALUES` = *string token*	Whether or not to prohibit automatic extension of the pointers (`yes`, `no`); default `no`
`RENAME` = *string token*	Whether to reset the default names of elements of the pointer if they do not have their own identifiers (`yes`, `no`); default `no`
`MODIFY` = *string token*	Whether to modify (instead of redefining) existing structures (`yes`, `no`); default `no`
`IPRINT` = *string tokens*	Information to be used by default to identify the pointers in output (`identifier`, `extra`); if this is not set, they will be identified in the standard way for each type of output
`EXTEND` = *string token*	Whether to extend (instead of redefining) an existing pointer (`yes`, `no`); default `no`

Parameters

`IDENTIFIER` = *identifiers*	Identifiers of the pointers
`VALUES` = *pointers*	Values for each pointer
`EXTRA` = *texts*	Extra text associated with each identifier

Description

Lists of data structures can be stored in a Genstat *pointer* structure to save having to type the list in full every time it is used. For example

```
POINTER [VALUES=Rain,Temp,Windspeed] Vars
VARIATE #Vars
READ [CHANNEL=2] #Vars
PRINT #Vars; DECIMALS=2,1,2
```

defines `Rain`, `Temp` and `Windspeed` to be variates, and then reads and prints their values. When none of the structures in the list is itself a pointer, the substitution symbol (`#`) simply replaces the pointer by its values. If, however, there are pointers in the list, they too are substituted, as are any pointers to which they point. An example is given below.

The individual elements of a pointer can also be referred to by the use of *suffixes*. We can refer to `Rain` above either using its own identifier, or as the first element of `Vars` by using the *suffix* `[1]`: so

`Vars[1]`	is	`Rain`
`Vars[2]`	is	`Temp`
`Vars[3]`	is	`Windspeed`

Furthermore, we can put a list within the brackets:

`Vars[3,1]`	is	`Windspeed,Rain`.

Also, you can put a null list to mean all the available suffixes of the pointer:

`Vars[]` is `Rain,Temp,Windspeed`.

Identifiers like `Vars[1]`, `Vars[2]` and `Vars[3]` are called *suffixed* identifiers and, in fact, you can use these even without defining the identifier of the pointer explicitly. Whenever a suffixed identifier is used, Genstat automatically sets up a pointer for the unsuffixed part of the identifier if it does not already exist. Furthermore the pointer will automatically be extended (whether it has been set up by you or by Genstat) if you later use a new suffix, like `Vars[93]` for example. Notice that the suffixes do not need to be a contiguous list, nor need they run from one upwards, although they must be integers; if you give a decimal number it will be rounded to the nearest integer (for example, −27.2 becomes −27).

The `SUFFIXES` option of the `POINTER` directive allows you to specify the required suffixes for pointers that are defined explicitly. For example

```
VARIATE [VALUES=1990,1991,1992,1993] Suffs
POINTER [NVALUES=4; SUFFIXES=Suffs] Profit
```

defines `Profit` to be a pointer of length four, with suffixes 1990 to 1993. If you are setting the suffixes explicitly, you might want to forbid Genstat to extend the pointer if another suffix is encountered later in the program; this can be done by setting option `FIXNVALUES=yes`.

We could actually omit the `NVALUES` option in the definition of the pointer `Profit` above, as Genstat can determine the length of the pointer by counting the number of values. However, by supplying a text instead of a scalar for `NVALUES` you can define *labels* for the suffixes of the pointer. The length of the text defines the number of values of the pointer, and its values give the labels. For example

```
TEXT [VALUES=name,salary,grade] Labs
POINTER [NVALUES=Labs] Employee
```

would allow you to refer to `Employee['name']`, `Employee['salary']`, and so on.

Usually, when the pointer is later used, Genstat requires the labels to be given exactly as in the definition. However, you can set option `CASE=ignored` to indicate that case is unimportant, so they can be specified in capitals, or lower case, or in any mixture. You can also set option `ABBREVIATE=yes` to allow each one to be abbreviated to the minimum number of characters required to distinguish it from the labels of earlier elements of the pointer.

The identifiers in a suffix list can be of scalars, variates or texts; this of course includes numbers and strings as unnamed scalars and texts respectively. If one of these structures contains several values, it defines a sub-pointer: for example `Vars[!(3,2)]` is a pointer with two elements, `Windspeed` and `Temp`. You must thus be careful not to confuse a sub-pointer with a list of some of the elements of a pointer: for example `Vars[!(3,2)]` is a single pointer with two elements, whereas `Vars[3,2]` is a list of the two structures `Windspeed` and `Temp`.

As mentioned above, a pointer can be extended automatically to include a new suffix, if that suffix is used with the pointer in your program. However, it is not possible to extend the pointer automatically to include a new label, as Genstat would not know which suffix to give it and an automatic choice could lead to errors or confusion. So, the `POINTER` directive has an option `EXTEND` which can be set to `yes` to do this explicitly. The pointer elements that are defined are then added to the existing elements of the pointer. So, we could add additional labels to the pointer `Employee`, above, by the statements

```
TEXT [VALUES=department,room] Newlabs
VARIATE [VALUES=4,5] Newsuffs
POINTER [NVALUES=Newlabs; SUFFIXES=Newsuffs; EXTEND=yes] Employee
```

adds `Employee['department']` as suffix 4, and `Employee['room']` as suffix 5. If you do not specify a label, the new suffix is still added (but unlabelled). If you do not specify a suffix, the new label is given a suffix of one plus the largest suffix already in the pointer. When `EXTEND=yes`, the `EXTRA` parameter and the `CASE`, `ABBREVIATE`, `FIXNVALUES`, `MODIFY` and

IPRINT options are ignored.

Elements of pointers can themselves be pointers, allowing you to construct trees of structures. For example

```
VARIATE A,B,C,D,E
POINTER R; VALUES=!P(D,E)
& S; VALUES=!P(B,C)
& Q; VALUES=!P(A,S)
& P; VALUES=!P(Q,R)
```

You can refer to elements within the tree by giving several levels of suffixes: for example P[2][1] is R[1] which is D; P[2,1][1,2] is (R,Q)[1,2] or D,E,A,S. The special symbol # allows you to list all the structures at the ends of the branches of the tree: #P replaces P by the identifiers of the structures to which it points (Q and R); then, if any of these is a pointer, it replaces it by its own values, and so on. Thus #P is the list A,B,C,D,E.

As you have seen, structures need not have an identifiers of their own, but may simply be identifiable as a member of a pointer using the suffix notation. Where a structure like this is a member of more than one pointer, Genstat will refer to it in output using the pointer with which it was first associated. So, for example, in

```
POINTER [NVALUES=2] P
& [VALUES=P[1,2],C] Q
VARIATE [VALUES=1,2,3,4] Q[]
PRINT Q[]
```

the output will be labelled as P[1], P[2] and C. However, we can set option RENAME=yes when Q is defined

```
POINTER [VALUES=P[1,2],C; RENAME=yes]
```

to request that the pointer Q takes precedence over earlier definitions, so the labels become Q[1], Q[2] and C.

Options: NVALUES, VALUES, SUFFIXES, CASE, ABBREVIATE, FIXNVALUES, RENAME, MODIFY, IPRINT, EXTEND.
Parameters: IDENTIFIER, VALUES, EXTRA.

See also

Directives: ASSIGN, STRUCTURE, DECLARE, DUMMY, LRV, SSPM, TSM, TREE.
Procedure: PDUPLICATE.
Functions: VSUMS, VTOTALS, VMEANS, VMEDIANS, VMINIMA, VMAXIMA, VRANGE, VCOVARIANCE, VCORRELATION, VSD, VSEMEANS, VSKEWNESS, VKURTOSIS, VVARIANCES, VNOBSERVATIONS, VNVALUES, VNMV, VPOSITIONS.
Genstat Reference Manual 1 Summary section on: Data structures.

PREDICT

Forms predictions from a linear or generalized linear model.

Options

PRINT = *string token*	What to print (description, lsd, predictions, se, sed, vcovariance); default desc, pred, se
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
COMBINATIONS = *string token*	Which combinations of factors in the current model to include (full, present, estimable); default esti
ADJUSTMENT = *string token*	Type of adjustment (marginal, equal); default marg
WEIGHTS = *table*	Weights classified by some or all of the factors in the model; default *
OFFSET = *scalar*	Value of offset on which to base predictions; default mean of offset variate
METHOD = *string token*	Method of forming margin (mean, total); default mean
ALIASING = *string token*	How to deal with aliased parameters (fault, ignore); default faul
BACKTRANSFORM = *string token*	What back-transformation to apply to the values on the linear scale, before calculating the predicted means (link, none); default link
SCOPE = *string token*	Controls whether the variance of predictions is calculated on the basis of forecasting new observations rather than summarizing the data to which the model has been fitted (data, new); default data
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, nonlinear); default *
DISPERSION = *scalar*	Value of dispersion parameter in calculation of s.e.s; default is as set in the MODEL statement
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default is as set in the MODEL statement
NBINOMIAL = *scalar*	Supplies the total number of trials to be used for prediction with a binomial distribution (providing a value *n* greater than one allows predictions to be made of the number of "successes" out of *n*, whereas the value one predicts the proportion of successes); default 1
PREDICTIONS = *table*, *scalar* or *pointer*	Saves predictions for each y variate; default *
SE = *table*, *scalar* or *pointer*	Saves standard errors of predictions for each y variate; default *
SED = *symmetric matrix* or *pointer*	Saves standard errors of differences between predictions for each y variate; default *
LSD = *symmetric matrix* or *pointer*	Saves least significant differences between predictions for each y variate (models with Normal errors only); default *
LSDLEVEL = *scalar*	Significance level (%) to use in the calculation of least significant differences; default 5
VCOVARIANCE = *symmetric matrix* or *pointer*	Saves variance-covariance matrices of predictions for

	each y variate; default *
SAVE = *identifier*	Specifies save structure of model from which to predict; default * i.e. that from latest model fitted

Parameters

CLASSIFY = *vectors*	Variates and/or factors to classify table of predictions
LEVELS = *variates*, *scalars* or *texts*	To specify values of variates, levels of factors
PARALLEL = *identifiers*	For each vector in the CLASSIFY list, allows you to specify another vector in the CLASSIFY list with which the values of this vector should change in parallel (you then obtain just one dimension in the table of predictions for these vectors)
NEWFACTOR = *identifiers*	Identifiers for new factors that are defined when LEVELS are specified

Description

The PREDICT directive can be used after the FIT directive to summarize the results of the regression, by using the fitted relationship to predict the values of the response variate at particular values of the explanatory variables. CLASSIFY, the first parameter of PREDICT, specifies those variates or factors in the current regression model whose effects you want to summarize. Any variate or factor in the current model that you do not include will be standardized in some way, as described below.

The LEVELS parameter specifies values at which the summaries are to be calculated, for each of the structures in the CLASSIFY list. For factors, you can select some or all of the levels, while for variates you can specify any set of values. A single level or value is represented by a scalar; several levels or values must be combined into a variate (which may of course be unnamed). Alternatively, if the factor has labels, you can use these to select the levels for the summaries by setting LEVELS to a text. A missing value in the LEVELS parameter is taken by Genstat to stand for all the levels of a factor, or for the mean value of a variate.

The PARALLEL parameter allows you to indicate that a factor or variate should change in parallel to another factor or variate. Both of these should have same number of values specified for it by the LEVELS parameter of PREDICT. The predictions are then formed for each corresponding set of values rather than for every combination of these values. For example, suppose we had fitted a quadratic model with explanatory variates X and Xsquared. We could then put

```
PREDICT Xsquared,X; PARALLEL=X,*;\
        LEVELS=!(0,4,16,36,64,100),!(0,2,4,6,8,10)
```

The PARALLEL parameter specifies that Xsquared should change in parallel to X, so that we obtain predictions only for matching values.

When you specify LEVELS, PREDICT needs to define a new factor to classify that dimension of the table. By default this will be an unnamed factor, but you can use the NEWFACTOR parameter to give it an identifier. The EXTRA attribute of the factor is set to the name of the corresponding factor or variate in the CLASSIFY list; this will then be used to label that dimension of the table of predictions.

You can best understand how Genstat forms predictions by regarding its calculations as consisting of two steps. The first step, referred to below as Step A, is to calculate the full table of predictions, classified by every factor in the current model. For any variate in the model, the predictions are formed at its mean, unless you have specified some other values using the LEVELS parameter; if so, these are then taken as a further classification of the table of predictions. The second step, referred to as Step B, is to average the full table of predictions over the classifications that do not appear in the CLASSIFY parameter: you can control the type of

averaging using the COMBINATIONS, ADJUSTMENT and WEIGHTS options. By default, the predictions are made at the mean of any offset variate, but option OFFSET can be used to specify another value at which the predictions should be made instead.

Printed output is controlled by settings of the PRINT option:

description	describes the standardization policies used when forming the predictions,
predictions	prints the predictions,
se	produces predictions and standard errors,
sed	prints standard errors for differences between the predictions,
lsd	prints least significant differences between the predictions (ordinary linear regression models or generalized linear models with the Normal distibution only), and
vcovariance	prints the variance and covariances of the predictions.

By default descriptions, predictions and standard errors are printed. The standard errors (and sed's) are relevant for the predictions when considered as means of those data that have been analysed, with the means formed according to the averaging policy defined by the options of PREDICT. The word *prediction* is used because these are predictions of what the means would have been if the factor levels been replicated differently in the data; see Lane & Nelder (1982) for more details. The LSDLEVEL option specifies the significance level (%) to use in the calculation of least significant differences (default 5%).

By default, the standard errors (and sed's) are not augmented by any component corresponding to the estimated variability of a new observation. However, you can set option SCOPE=new to request that the variance of predictions should be calculated on the basis of forecasting new observations rather than of summarizing the data to which the model has been fitted. This setting cannot be used if the predictions are to be standardized for the effects of any factors in the model; in other words, all factors in the current model must be listed in the CLASSIFY parameter of the PREDICT statement. In addition, it cannot be used when making predictions from generalized linear models with option BACKTRANSFORMATION=none, nor with weighted regression. The effect of SCOPE=new is to form variances for each predicted value by combining the variance of the estimated mean value of the prediction (as produced for SCOPE=data) together with the estimated variance of a new observation with the same values of explanatory variates and factors:

"new" variance = "data" variance + (dispersion × variance function)

The DISPERSION and DMETHOD options allow you to change the method by which the variance of the distribution of the response values is obtained for calculating the standard errors. These options operate like the corresponding options of MODEL (except that they apply only to the current statement). The default is to use the method as originally defined by the MODEL statement.

The NBINOMIAL parameter can be used to supply the total number of trials to be used for prediction with a binomial distribution when option BACKTRANSFORMATION is set to link. If you provide a value *n* greater than one, Genstat will predict the number of "successes" out of *n*. The default, NBINOMIAL=1, causes Genstat to predict the proportion of successes.

You can send the output to another channel, or to a text structure, by setting the CHANNEL option.

The COMBINATIONS option specifies which cells of the full table in Step A are to be filled for averaging in Step B. The default, COMBINATIONS=estimable, uses all the cells other than those that involve parameters that cannot be estimated, for example because of aliasing. Alternatively, you can set COMBINATIONS=present to exclude cells for factor combinations that do not occur in the data, or COMBINATIONS=full to use all the cells. When COMBINATIONS=estimable or COMBINATIONS=present the LEVELS parameter is overruled.

Any subsets of factor levels in the LEVELS parameter are ignored, and predictions are formed for all the factor levels that occur in the data or are estimable. Likewise, the full table cannot then be classified by any sets of values of variates; the LEVELS parameter must then supply only single values for variates.

The ADJUSTMENT and WEIGHTS options define how the averaging is done in Step B. Values in the full table produced in Step A are averaged with respect to all those factors that you have not included in the settings of the CLASSIFY parameter. By default, the levels of any such factor are combined with what we call *marginal weights*: that is, by the number of occurrences of each of its levels in the whole dataset. The ADJUSTMENT and WEIGHTS options allow you to change the weights. The setting ADJUSTMENT=equal specifies that the levels are to be weighted equally. (This corresponds to the default weighting used by VPREDICT.) The WEIGHTS option is more powerful than the ADJUSTMENT option, allowing you to specify an explicit table of weights. This table can be classified by any, or all, of the factors over whose levels the predictions are to be averaged; the levels of remaining factors will be weighted according to the ADJUSTMENT option. Moreover, you can classify the weights by the factors in the CLASSIFY parameter as well, to provide different weightings for different combinations of levels of these factors. If you supply explicit weights in the WEIGHTS option, any setting of the COMBINATIONS option is ignored. You will find explicit weights useful in particular when you have population estimates of the proportions of each level of a factor – proportions which may not be matched well in the available data.

If a model contains any aliased parameters, predicted values cannot be formed for some cells of the full table without assuming a value for the aliased parameters. With the default setting, COMBINATIONS=estimable, no predictions are formed for these cells. When COMBINATIONS=full, if the aliased parameters simply represent effects of variates that are correlated with other explanatory variables in the model, it may be sufficient just to ignore them. This can be done by setting the ALIASING option to ignore. The aliased parameters are then taken to be zero, and fitted values are calculated for all cells of the table from the remaining parameters in the model. Aliasing can also occur if there are some combinations of factors that do not occur in the data, and here it may be more sensible to set option COMBINATIONS=present so that these cells are all excluded from the calculation of predictions. The final way to overcome aliasing is to supply explicit weights using the WEIGHTS option.

Averaging is usually the appropriate way of combining predicted values over levels of a factor. But sometimes summation is needed, for example in the analysis of counts by log-linear models. You can achieve this by setting the METHOD option to total. The rules about weights and so on still apply. In a generalized linear model, averaging is done by default on the scale of the original response variable, not on the scale transformed by the link function. In other words, linear predictors are formed for all the combinations of factor levels and variate values specified by PREDICT, and then transformed by the link function back to the natural scale. This back-transformation may be useful when you are reporting results, since the tables from PREDICT can then be interpreted as natural averages of means predicted by the fitted model. You can set option BACKTRANSFORM=none if you want the averaging to be done on the scale of the linear predictor; PREDICT will then form averages and report predictions on the transformed scale.

PREDICT calculates the standard errors of predictions from iterative models by using first-order approximations that allow for the effect of the link function. Thus you should interpret them only as a rough guide to the variability of individual predictions.

The PREDICTIONS, SE, SED, LSD and VCOVARIANCE options let you save the results of PREDICT as well as, or instead of, printing them. They are saved in a single data structure if there is only one y-variate, or a pointer to several if there is more than one.

The SAVE option allows you to specify the regression save structure of the analysis on which the predictions are based. If SAVE is not set, the most recent regression model is used.

The NOMESSAGE option controls printing of messages. The nonlinear setting suppresses messages about the approximate nature of standard errors of predictions in generalized linear models, and the dispersion setting prevents reminders appearing about the basis of the standard errors.

Options: PRINT, CHANNEL, COMBINATIONS, ADJUSTMENT, WEIGHTS, OFFSET, METHOD, ALIASING, BACKTRANSFORM, SCOPE, NOMESSAGE, DISPERSION, DMETHOD, NBINOMIAL, PREDICTIONS, SE, SED, LSD, LSDLEVEL, VCOVARIANCE, SAVE.
Parameters: CLASSIFY, LEVELS, PARALLEL, NEWFACTOR.

Reference

Lane, P.W. & Nelder, J.A. (1982). Analysis of covariance and standardization as instances of prediction. *Biometrics*, **38**, 613-621.

See also

Directives: FIT, RDISPLAY, VPREDICT.
Procedure: HGPREDICT.
Genstat Reference Manual 1 Summary section on: Regression analysis.

PRINT

Prints data in tabular format in an output file, unformatted file or text.

Options

`CHANNEL` = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
`SERIAL` = *string token*	Whether structures are to be printed in serial order, i.e. all values of the first structure, then all of the second, and so on (`yes`, `no`); default `no`, i.e. values in parallel
`IPRINT` = *string tokens*	What identifier and/or text to print for the structure (`identifier`, `extra`, `associatedidentifier`), for a table `associatedidentifier` prints the identifier of the variate from which the table was formed (e.g. by `TABULATE`), `IPRINT=*` suppresses the identifier altogether; default `iden`
`RLPRINT` = *string tokens*	What row labels to print (`labels`, `integers`, `identifiers`), `RLPRINT=*` suppresses row labels altogether; default `labe`, `iden`
`CLPRINT` = *string tokens*	What column labels to print (`labels`, `integers`, `identifiers`), `CLPRINT=*` suppresses column labels altogether; default `labe`, `iden`
`RLWIDTH` = *scalar*	Field width for row labels; default 13
`INDENTATION` = *scalar*	Number of spaces to leave before the first character in the line; default 0
`WIDTH` = *scalar*	Last allowed position for characters in the line; default width of current output file
`SQUASH` = *string token*	Whether to omit blank lines in the layout of values (`yes`, `no`); default `no`
`MISSING` = *text*	What to print for missing value; default uses `'*'` for numbers and blanks in texts
`ORIENTATION` = *string token*	How to print vectors or pointers (`down`, `across`); default `down`, i.e. down the page
`ACROSS` = *scalar* or *factors*	Number of factors or list of factors to be printed across the page when printing tables; default for a table with two or more classifying factors prints the final factor in the classifying set and the notional factor indexing a parallel list of tables across the page, for a one-way table only the notional factor is printed across the page
`DOWN` = *scalar* or *factors*	Number of factors or list of factors to be printed down the page when printing tables; default is to print all other factors down the page
`WAFER` = *scalar* or *factors*	Number of factors or list of factors to classify the separate "wafers" (or slices) used to print the tables; default 0
`PUNKNOWN` = *string token*	When to print unknown cells of tables (`present`, `always`, `zero`, `missing`, `never`); default `pres`
`UNFORMATTED` = *string token*	Whether file is unformatted (`yes`, `no`); default `no`
`REWIND` = *string token*	Whether to rewind unformatted file before printing (`yes`, `no`); default `no`
`WRAP` = *string token*	Whether to wrap output that is too long for one line onto subsequent lines, rather than putting it into a subsequent

	"block" (`yes`, `no`); default `no`
`STYLE` = *string token*	Style to use for an output file (`plaintext`, `formatted`); default `*` uses the current style of the channel
`PMARGIN` = *string tokens*	Which margins to print for tables (`full`, `columns`, `rows`, `wafers`); default `full`
`OMITMISSINGROWS` = *string token*	Whether to omit rows of tables that contain only missing values (`yes`, `no`); default `no`
`VSPECIAL` = *scalar* or *variate*	Special values to be modified in the output
`TSPECIAL` = *text*	Strings to be used for the special values; must be set if `VSPECIAL` is set

Parameters

`STRUCTURE` = *identifiers*	Structures to be printed
`FIELDWIDTH` = *scalars*	Field width in which to print the values of each structure (a negative value *-n* prints numbers in E-format in width *n*); if omitted, a default is determined (for numbers, this is usually 12; for text, the width is one more character than the longest line)
`DECIMALS` = *structures*	Number of decimal places for numerical data structures, a scalar if the same number of decimals is to be used for all values of the structure, or a data structure of the same type and size to use different numbers of decimals for each value; if omitted or set to a missing value, a default is determined which prints the mean absolute value to 4 significant figures
`CHARACTERS` = *scalars*	Number of characters to print in strings
`SKIP` = *scalars* or *variates*	Number of spaces to leave before each value of a structure (`*` means a new line before structure)
`FREPRESENTATION` = *string tokens*	How to represent factor values (`labels`, `levels`, `ordinals`); default is to use `labels` if available, otherwise `levels`
`JUSTIFICATION` = *string tokens*	How to position values within the field (`right`, `left`, `center`, `centre`); if omitted, `right` is assumed
`MNAME` = *string tokens*	Name to print for table margins (`margin`, `total`, `nobservd`, `mean`, `minimum`, `maximum`, `variance`, `count`, `median`, `quantile`); if omitted, "Margin" is printed
`DREPRESENTATION` = *scalars* or *texts*	Format to use for dates and times (stored in numerical structures)
`HEADING` = *texts*	Heading to be used for vectors printed in columns down the page; default is to use the information requested by the `IPRINT` option
`TLABELS` = *texts*	If this is specified for a table `STRUCTURE`, the values of the table are interpreted as references to lines within the `TLABELS` text that are to be printed instead of the values of the table itself

Description

The contents of Genstat data structures can be displayed, with appropriate labelling, using the PRINT directive. Output can be printed in the current output channel, or sent to other channels, or put into a text structure. PRINT has many options and parameters to allow you to control the style and format of the output but, in most cases, these can be left with their default settings.

For a quick display of the contents of a list of data structures, you need only give the name of the directive, PRINT, and then list their identifiers. For example

```
PRINT Source,Amount,Gain
```

The output is fully annotated with the identifiers, and with row and column labels or numbers, where appropriate. Factors are represented by their labels if available, and otherwise by their levels. The layout of the values is determined automatically by the size and shape of the structures to be printed, and by the space needed to print individual values. The output is arranged in columns; the structures are split if the page is not wide enough, so that one set of columns is completed before the next is printed.

With vectors, the default is to print their values in parallel. Alternatively, you can request that structures are printed in series, one below another, by setting option SERIAL=yes. Of course, if the structures to be printed have different shapes or sizes, their values can be printed only in series. The setting SERIAL=no is then ignored except that, to save space, any vectors or pointers are then printed across the page (that is, as though you had set ORIENTATION=across).

Genstat annotates each set of values by the identifier of the structure (but this can be controlled by the IPRINT option or the HEADING parameter described below), and it automatically chooses a suitable format. For a numerical structure, the default is to use a field of *f* characters. Generally, the value of *f* is 12, but another value can be defined using the FIELDWIDTH option of the SET directive. If the DECIMALS parameter was set when the structure was declared, this will define the number of decimal places in the output; otherwise, the number of decimal places is determined by calculating the number that would be required to print its mean absolute value to at least *d* significant figures. Generally, *d* is four, but this can be redefined using the SIGNIFICANTFIGURES option of the SET directive. Texts (and labels of factors) are usually printed in a field of *f* characters but this is extended if any of the strings in the text requires a wider field. You can define your own formats using the parameters FIELDWIDTH, DECIMALS, CHARACTERS, SKIP and JUSTIFICATION.

FIELDWIDTH and DECIMALS both operate in a straightforward way. The only potential complication is that a negative FIELDWIDTH can be used to print numbers in scientific format (for example 7.3 E1 instead of 73), with DECIMALS significant places. The DECIMALS parameter is ignored for strings, like the labels of the factors Source and Amount. For a numerical data structure, you can either set DECIMALS to a scalar to use the same number of decimals for all the values of the structure. Alternatively, if you want to use a different number of decimals for each value, you can supply a data structure of the same type and size as STRUCTURE. If DECIMALS is omitted or if it contains a missing value, a default is used which prints the mean absolute value to *d* significant figures, as above.

In the same way, the CHARACTERS parameter is ignored for numbers; for strings, it allows you to control the number of characters that are printed. By default, Genstat prints all the characters in each string of a text or factor label, unless the CHARACTERS parameter was set to a lesser number when the text or factor was declared.

Textual strings can contain typesetting commands to represent Greek and special symbols. (These are most useful in PRINT, but can be used in any directive that generates output.) The commands are converted automatically by Genstat to match the style of output (HTML, LaTeX, plain-text or RTF). The commands are all introduced by the character tilde (~). So, to use tilde as an ordinary character, you need to specify the special symbol ~{~} as defined below.

If Genstat finds a mistake in the syntax of a command, it will not issue a failure diagnostic but will output the remainder of the string (including any commands) as plain text. So, for example,

legacy code containing tilde characters should continue to generate the output in its previous form. The following comands define Greek characters and various special symbols.

~{~}	tilde symbol; also see ~{tilde}
~{alpha}	Greek character alpha
~{beta}	Greek character beta
~{gamma}	Greek character gamma
~{delta}	Greek character delta
~{epsilon}	Greek character epsilon
~{varepsilon}	Greek character epsilon (variant)
~{zeta}	Greek character zeta
~{eta}	Greek character eta
~{theta}	Greek character theta
~{vartheta}	Greek character theta (variant)
~{iota}	Greek character iota
~{kappa}	Greek character kappa
~{lambda}	Greek character lambda
~{mu}	Greek character mu
~{nu}	Greek character nu
~{xi}	Greek character xi
~{omicron}	Greek character omicron
~{pi}	Greek character pi
~{varpi}	Greek character pi (variant)
~{rho}	Greek character rho
~{varrho}	Greek character rho (variant)
~{sigma}	Greek character sigma
~{varsigma}	Greek character sigma (terminal version)
~{tau}	Greek character tau
~{upsilon}	Greek character upsilon
~{phi}	Greek character phi
~{varphi}	Greek character phi (variant)
~{chi}	Greek character chi
~{psi}	Greek character psi
~{omega}	Greek character omega
~{bull} or ~{bullet}	bullet
~{cdot}	decimal point; also see ~{middot}
~{div} or ~{divide}	divide symbol
~{gg}	">>" symbol; also see ~{raquo}
~{laquo}	"<<" symbol; also see ~{ll}
~{ll}	"<<" symbol; also see ~{laquo}
~{middot}	alternative way of specifying a decimal point; also see ~{cdot}
~{minus}	minus symbol
~{plusminus}	"+ or minus" symbol; also see ~{pm}
~{pm}	"+ or minus" symbol; also see ~{plusminus}
~{raquo}	">>" symbol; also see ~{gg}
~{sqrt}	square-root symbol
~{oplus}	+ within circle
~{ominus}	minus symbol within circle
~{otimes}	multiply symbol within circle
~{oslash}	slash symbol within circle
~{odot}	dot within circle

`~{tilde}`	tilde symbol; also see ~{~}
`~{times}`	multiply symbol
`~{break}`	starts a new line in captions

The character definitions (within the curly brackets) can be abbreviated. Genstat checks through the possibilities, in the order defined above, until it finds the first match. Greek characters in capital letters can be obtained by beginning the name of the character with a capital letter, for example `~{Sigma}`; subsequent capital letters are irrelevant.

The style of font can be changed to bold or italic.

`~bold` or `~b`	introduces a sequence of bold characters; these must be placed within curly brackets and any spaces between `~bold` and the opening curly bracket are ignored. e.g. `~bold {Please note:}`
`~italic` or `~i`	introduces a sequence of italic characters; these must be placed within curly brackets and any spaces between `~italic` and the opening curly bracket are ignored. e.g. `~italic {Passer domesticus}`

You can also produce output in the same style as Genstat uses when it echoes commands in the output.

`~genstat` or `~g`	introduces some output in the style that Genstat uses to echo commands; it must be placed within curly brackets and any spaces between `~genstat` and the opening curly bracket are ignored.

You can define subscripts and superscripts to use, for example, in equations.

`~_`	introduces a subscript; if the subscript is a single character it can be placed immediately after _, otherwise it must be placed within curly brackets; any spaces between ~_ and the opening curly bracket are ignored.
`~^`	introduces a superscript; if the superscript is a single character it can be placed immediately after ^, otherwise it must be placed within curly brackets; any spaces between ~^ and the opening curly bracket are ignored.

You can use special characters in subscripts or superscripts, but fonts must be specified outside the subscript or superscript. For example:

`~i {x~_{i,j}}`	defines $x_{i,j}$,
`x~^ {2n}`	defines x^{2n}, and
`~i{x~_{i,j}}~^2`	defines $x_{i,j}^{\ 2}$
`~b{X}~i{~_{i,j}}~^2`	defines $\mathbf{X}_{i,j}^{\ 2}$.

For additional flexibility, you can specify output information in either HTML, LaTeX or RTF. This will be inserted only into output constructed by Genstat in the same style. You can also supply information to be included only in plain-text output (which may, for example, be your translation of the HTML, LaTeX or RTF information).

`~html` or `~h`	introduces a sequence of information in HTML; the information must be placed within curly brackets and any spaces between `~html` and the opening curly bracket are ignored.
`~latex` or `~l`	introduces a sequence of information in LaTeX; the information must be placed within curly brackets and any spaces between `~latex` and the opening curly bracket are ignored. The information may itself contain curly brackets. These are assumed to be paired according to the usual rules of LaTeX, except that any curly brackets preceded by

	the LaTeX escape character \ are ignored.
~plain or ~p	introduces a sequence of information to be inserted only in plain-text output; the information must be placed within curly brackets and any spaces between ~plain and the opening curly bracket are ignored.
~rtf or ~r	introduces a sequence of information in RTF; the output must be placed within curly brackets and any spaces between ~rtf and the opening curly bracket are ignored. The information may itself contain curly brackets. These are assumed to be paired according to the usual rules of RTF, except that curly brackets preceded by the RTF escape character \ are ignored.

The DREPRESENTATION parameter is used to specify how to print numbers that represent dates and times. The DECIMALS parameter is then ignored. The setting of DREPRESENTATION is either a scalar indicating a predefined format, or a string defining a custom format. The string for a custom format contains a sequence of keys to represent the required components of the date and time. The available keys are:

d	day number within the month, using the minimum number of digits (e.g. 3, 12)
dd	day number within the month, using two digits (e.g. 03, 12)
dth	day number with one digit and suffix (e.g. 3rd, 12th)
m	month number, using the minimum number of digits
mm	month number, using two digits
mmm	abbreviated month name (Jan, Feb, Mar, Apr, May, June, July, Aug, Sept, Oct, Nov, Dec)
mmmm	month name in full
yy	year as a two-digit number (omitting the century)
yyyy	year as a four-digit number (including the century)
weekday	day of the week (Monday, Tuesday, and so on)
wday	abbreviated day of the week (Mon, Tue, Wed, Thur, Fri, Sat, Sun)
time24	time, including seconds, using a 24 hour clock
time12	time, including seconds, using a 12 hour clock (with a.m. and p.m.)
time100	time, using 24 hour clock and including hundredths of seconds
hours	elapsed time in hours, minutes and seconds
hours100	elapsed time in hours, minutes, seconds and hundredths of seconds
minutes	elapsed time in minutes and seconds
minutes100	elapsed time in minutes, seconds and hundredths of seconds
seconds	elapsed time in seconds
seconds100	elapsed time in seconds and hundredths of second

You can also insert one or more separators between the keys, any combination of space ( ), slash (/), hyphen (-) and comma (,).

Note: the operation of the 2-digit representation of a year is controlled by a "break point". This has the initial default of 30, but that can be changed by the YEAR2DIGITBREAK option of SET, or in the DateFormat tab of the Options menu in Genstat *for Windows*. With the initial default of 30, for example, years from 1930 to 2029 will be represented as two digits, but others will be printed with four digits.

To simplify the specification of the most commonly used formats, a range of standard predefined formats are available. These are specified by supplying a scalar containing one of the numerical codes in the left-hand column of the table below.

code	format	example
1	dd/mm/yy	03/08/98
2	dd/mm/yyyy	03/08/1998
3	d/m/yy	3/8/98
4	d/m/yyyy	3/8/1998
5	ddmmyy	030898
6	ddmmyyyy	03081998
7	ddmmmyy	03Aug98
8	ddmmmyyyy	03Aug1998
9	dd-mmm-yy	03-Aug-98
10	dd-mmm-yyyy	03-Aug-1998
11	dmmmyy	3Aug98
12	dmmmyyyy	3Aug1998
13	d-mmm-yy	3-Aug-98
14	d-mmm-yyyy	3-Aug-1998
15	d-mmmm-yy	3-August-98
16	d-mmmm-yyyy	3-August-1998
17	yymmdd	980803
18	yyyymmdd	19980803
19	yy/mm/dd	98/08/03
20	yyyy/mm/dd	1998/08/03
21	mmddyy	080398
22	mmddyyyy	08031998
23	mm/dd/yy	08/03/98
24	mm/dd/yyyy	08/03/1998
25	mmm-dd-yy	Aug-03-98
26	mmm-dd-yyyy	Aug-03-1998
27	yyyy-mm-dd	1998-08-03
28	weekday, dth mmmm yyyy	Monday, 3rd August 1998
29	weekday	Monday
30	mmm-yy	Aug-98
31	yy	98
32	yyyy	1998
33	dd-mmm-yyyy time100	03-Aug-1998 18:55:30.35
34	yyyy-mm-dd time (ODBC Standard format)	1998-08-03 18:55:30
35	dd-mmm-yyyy time12	03-Aug-1998 6:55:30 pm
36	time24	18:55:30
37	time12	6:55:30 pm

38	hours	48:55:30
39	seconds	68538.35
40	dd/mm/yyyy time24	03/08/1998 18:55:30
41	m-yy	8-98
42	m-yyyy	8-1998
43	mm-yy	08-98
44	mm-yyyy	08-1998
45	d/m	3/8
46	dd/mm	03/08
47	d-mmm	8-Aug
48	dd-mmm	08-Aug
49	mmm	Aug

You can also use the custom date formats supported by the client in Genstat *for Windows*. See the `Date Formats` page in the on-line help for details.

The `SKIP` parameter allows you to place extra spaces between the values of each structure. By default, no extra spaces are inserted unless a value fills the field completely, when a single space will be inserted; there is also a blank line before the first printed line. `SKIP` can be set to either a scalar or a variate in which a positive integer *n* requests that *n* spaces are left and a missing value can be used to request a blank line.

The values can be left-justified by setting the `JUSTIFICATION` parameter to `left`, or centred by setting it to `center` or `centre`.

The `FREPRESENTATION` parameter controls the printing of the factor values. By default Genstat will print labels if there are any; if there are none, it prints the levels. The `ordinals` setting represents the values by the integers 1 upwards.

The `ORIENTATION` option is relevant only when you are printing vectors or pointers. By setting `ORIENTATION=across`, the values are printed in alternate lines, across the page. To ensure that these line up correctly, the fieldwidth is taken as the maximum of those specified for the printed structures, while the field used to print their identifiers is given by the `RLWIDTH` option (by default 13).

When there is too much output to fit across the page, Genstat will print the output in more than one block unless option `WRAP` is set to `yes`. Then Genstat simply wraps each line onto subsequent lines. This is likely to be useful mainly if you are printing the contents of the structures to be read by another program. You might then also wish to suppress the identifiers by setting option `IPRINT=*` and remove blank lines by setting option `SQUASH=yes`.

The default option setting, `IPRINT=identifier`, will label the output with the identifier of the structure. Putting `IPRINT=identifier,extra` will also include any text that has been associated with the structure by the `EXTRA` parameter when it was declared, while the setting `associatedidentifier` can be used when a table has been produced by the `TABULATE` and `AKEEP` directives, to request that the output be labelled with the identifier of the variate from which the table was formed.

If you are printing vectors in parallel columns down the page, you can use the `HEADING` parameter to specify a text for each vector. This will then be used as a heading for that column, instead of the information requested by `IPRINT`. Note, though, that setting `IPRINT=*` will suppress any heading texts of the vectors as well as their identifiers.

The width of each line can be controlled by the `WIDTH` option; the default is to take the full available width. The `INDENTATION` option specifies the number of spaces to leave before each line; by default there are none.

The `CHANNEL` option determines where the output appears. By default, the output is placed

in the current output channel, but CHANNEL can be set to a scalar to send it to another output channel; the correspondence between channels and files on the computer is explained in the description of the OPEN directive. Alternatively, you can set CHANNEL to the identifier of a text to store the output. The text need not be declared in advance; any undeclared structure that is specified by CHANNEL will be defined automatically as a text. Each line of output becomes one value of the text and if the text already has values they will be replaced. You are most likely to want to do this in order to manipulate the text further. Remember, however, that if you print the text later on, its strings will be right-justified by default, so you will need to set JUSTIFICATION=left in the later PRINT statement to achieve the normal appearance of your output. The maximum (and default) line length of this text is the length of what is called the *output buffer*. This is likely to be 200 on most computers. If you intend to print it to an output file, you should set the WIDTH option as appropriate.

The MISSING option allows you to specify a string to represent missing values, instead of the default that uses the asterisk symbol for missing numbers, and blanks for missing values in texts. For example, you could set MISSING='unknown' or MISSING=' '.

The VSPECIAL and TSPECIAL options allow you to substitute textual strings for other values of numerical structures. The values are specified, in either a scalar or a variate, using the VSPECIAL option. The TSPECIAL option then specifies a text, with as many values as the VSPECIAL scalar or variate, to define the strings to be printed instead. For example, in the following program, values of prob less than 0.001 are set to −1, and then printed as '<0.001'.

```
CALCULATE prob = prob * (prob .GE. 0.001) - (prob .LT. 0.001)
PRINT [VSPECIAL=-1; TSPECIAL='<0.001'] prob
```

PRINT can easily be used to print matrices and tables, by taking the default layout and labelling. For tables with more than one dimension, the usual layout has one factor across the page and the others down the page; tables with only one dimension are printed down the page. Several tables can be printed in parallel, provided they all have the same classifying factors. The tables are then printed in alternate columns, as though they formed a larger table with an extra factor (called the table-factor) representing the list of tables. This extra factor thus becomes another (in fact, the final) factor to be printed across the page.

This default layout can be changed using the ACROSS, DOWN and WAFER options. You may wish to do this simply by changing the factors which appear down and across the page. The ACROSS option can be set to a scalar to specify how many factors should be printed across the page, or to a list of factors to say which ones they should be. DOWN similarly specifies the factors to be printed down the page. However, you cannot specify a list of factors for one of these options and a scalar for any of the others. The table-factor can be represented in these lists by inserting a * in the required position; if you do not mention the table-factor in either list it remains as the last factor in the ACROSS list.

The WAFER option allows you to split the output up into subtables or "wafers". This is particularly useful if the tables have many classifying factors, or if the factors have very long labels. The setting can again be either a scalar or a list of factors (possibly including the table-factor). Each subtable has a heading indicating its position in the full table. If the table-factor is included in the wafer, the identifier of the appropriate table will be printed at the beginning of the label for that wafer; this does not mean that the table-factor itself has been moved, simply that the labelling has been rearranged to make it easier to read.

You need not specify all the options DOWN, ACROSS and WAFER. If you leave any of them out PRINT will deduce the missing information.

When a table has margins, usually they will all be printed. However, you can control which are printed, by specifying the following settings of the PMARGIN option:

full	print all margins (default),
columns	print margins over column factors,
rows	print margins over row factors, and

`wafers`	print margins over wafer factors.

The `OMITMISSINGROWS` option also operates only on tables; if you set it to `yes`, `PRINT` will omit any lines of output where the tables contain only missing values.

You can control the space allowed for labels of the `DOWN` factors by using the `RLWIDTH` option. By default this is set to 13, but you might want something else if the labels are very small. If the width provided (by you, or implicitly) is inadequate, `PRINT` automatically resets it to accommodate the longest row label. The labelling of rows by the down factors is controlled by the `RLPRINT` option. The default, `RLPRINT=labels,identifiers`, prints the identifiers of the factors and their levels or labels. Similarly, the `CLPRINT` controls the labelling of columns by the across factors.

When tables are produced by `TABULATE` Genstat sets an internal indicator for use by `PRINT` to indicate the appropriate label for any margins. When a single table is printed this name will be used by default. When printing tables in parallel, if they all have the same setting of the margin name indicator, the appropriate name is used. If they have different settings, or none at all (tables from sources other than `TABULATE`) the margins will be labelled `Margin` by default. You can change the label by setting the `MNAME` parameter. Tables printed in parallel must have the same label throughout, and Genstat will take the one specified for the first table in the list. But in serial printing, you can use a different margin name for each table.

The `TABULATE` and `AKEEP` directives also record the identifier of the variate from which the table was formed, and you can request that this be used to label the output, instead of the identifier of the table itself, by setting the `IPRINT` option to `associatedidentifier`.

The `PUNKNOWN` option controls the printing of the "unknown" cell of a table. The default action is to print this cell, labelled with the table identifier, but only if it contains a value other than missing value or zero. You can select one of five settings:

`present` (default)	print value if not missing or zero
`always`	print the unknown cell regardless of value
`zero`	print unless the value is zero
`missing`	print unless the value is missing
`never`	do not print the unknown cell whatever its value

Genstat tables can only contain numbers. However, you can use the `TLABELS` parameter to print tables of textual strings. You first need to form a Genstat text structure containing all the strings that may occur. Then form a table with the required classifying factors and, in each cell of the table, put the number of the line (within the text) of the string that you want to print there. For example

```
FACTOR [LABELS=!t(April,May,June)] Month
TEXT   [VALUES=North,South,East,West] Direction
TABLE  [CLASSIFICATION=Month; VALUES=12.2,5.8,10.7] MeanSpeed
&      [VALUES=4,2,4] MainDirection
PRINT  MeanSpeed,MainDirection; TLABELS=*,Direction
```

will print `'West'` in the first and third cells of the table for `MainDirection` (April and June), and `'South'` in the second cell (May).

Options `ACROSS`, `DOWN`, `WAFER`, `RLPRINT` and `CLPRINT` also apply to matrices. By default, though, if you have several matrices they will be printed one after another on the page.

With symmetric matrices the only options of these that are relevant are `RLPRINT` and `CLPRINT`; a further setting `integer` is available for these to request that the rows or columns be labelled by the integers 1 onwards, as well as, or instead of the labels provided with the symmetric matrix: for example setting `RLPRINT=integers` and `CLPRINT=integers, labels` would identify the rows by integers and the columns with integers and labels.

The `UNFORMATTED` option can be used to send output to unformatted files. These can store values of data structures, so that they can later be input again using `READ`. This provides a convenient of way to free some space temporarily. It can also save computing time if you have

a large data set that may need to be read several times. Input from character files is slow. So after vetting a large data set, it will be read more efficiently on future occasions if you transfer its contents to an unformatted file. As an alternative you could use backing store, but this stores the attributes of the structures as well as their values, and so access will take longer. You can also use these facilities to transfer data between Genstat and other programs. The only other options that are relevant to unformatted files are CHANNEL, REWIND and SERIAL. Genstat automatically creates an *unformatted workfile*, on channel 0, to which unformatted output is sent by default (by PRINT), and from which unformatted input is taken by default (by READ). This file is deleted automatically at the end of a Genstat run. It is usually quicker to read and write structures in series. Also the values of the structures transferred in parallel must all be of the same *mode*. Neither texts nor factors can be stored in parallel with values of the other, numerical, structures: scalars, variates, matrices or tables. As an example, we first open a file, and declare some variates, matrices and factors.

```
OPEN 'BDAT'; CHANNEL=3; FILETYPE=unformatted
VARIATE X,Y,Z; VALUES=!(11...19),!(21...29),!(31...39)
MATRIX [ROWS=2; COLUMNS=3; VALUES=11,12,13,21,22,23] M
FACTOR [LEVELS=3; VALUES=1,3,2,3,1,2,2,2,1,3] F
```

The next three statements store data for M and F on the file named BDAT and data for X, Y and Z (in parallel) on the workfile.

```
PRINT [CHANNEL=3; SERIAL=yes; UNFORMATTED=yes] M,F
PRINT [UNFORMATTED=yes] X,Y,Z
```

You can now free the space for numerical data for other purposes, by putting

```
DELETE X,Y,Z,F,M
```

By rewinding the files we can read the data back into Genstat.

```
READ [UNFORMATTED=yes; REWIND=yes] X,Y,Z
READ [CHANNEL=3; SERIAL=yes; UNFORMATTED=yes; REWIND=yes] M,F
```

You can also re-use the external file BDAT in a later job. If you change the lengths of structures, you must remember to reset them to their original values before you use unformatted READ to recover the data values from the file. Only the data values are stored in unformatted files, and not the attributes (such as lengths) as in backing-store files.

The style in which the output is generated will depend on how the channel has been opened (see the OPEN directive). An ordinary output file (i.e. one with option UNFORMATTED=no) may have been opened to take output as either plain text, HTML (as used for example by web browsers), RTF (as used by word processors such as Microsoft Word) or LaTeX. Plain-text output assumes that all characters occupy an equal width, so columns are aligned by use of space characters. The other styles use special codes to define the columns. However, you can set option STYLE=plain to request that output to files with these other styles should use spaces instead. This is useful particularly in procedures, when you may want to print a "sentence" containing information from several different data structures. The output of blank lines is still controlled by the SQUASH option.

Options: CHANNEL, SERIAL, IPRINT, RLPRINT, CLPRINT, RLWIDTH, INDENTATION, WIDTH, SQUASH, MISSING, ORIENTATION, ACROSS, DOWN, WAFER, PUNKNOWN, UNFORMATTED, REWIND, WRAP, STYLE, PMARGIN, OMITMISSINGROWS, VSPECIAL, TSPECIAL.

Parameters: STRUCTURE, FIELDWIDTH, DECIMALS, CHARACTERS, SKIP, FREPRESENTATION, JUSTIFICATION, MNAME, DREPRESENTATION, HEADING.

Action with RESTRICT

You can restrict any vector (variate, factor or text) to specify that only a subset of its units should be printed. When printing in series the vectors can be restricted to different subsets; but with parallel printing any restriction is applied to all the vectors (and any pointers) so, if more than

one vector is restricted, they must all be restricted in the same way.

See also
Directives: CAPTION, COPY, OPEN, PAGE, SKIP, DUMP, LIST.
Procedures: DECIMALS, MINFIELDWIDTH.
Genstat Reference Manual 1 Summary section on: Input and output.

PROCEDURE

Introduces a Genstat procedure.

Options

PARAMETER = *string token*	Whether to process the structures in each parameter list of the procedure sequentially using a dummy to store each one in turn, or whether to put them all into a pointer so that the procedure is called only once (`dummy`, `pointer`); default `dumm`
RESTORE = *string tokens*	Which aspects of the Genstat environment to store at the start of the procedure and restore at the end (`inprint`, `outprint`, `outstyle`, `diagnostic`, `errors`, `pause`, `prompt`, `newline`, `case`, `run`, `units`, `blockstructure`, `treatmentstructure`, `covariate`, `asave`, `dsave`, `msave`, `rsave`, `tsave`, `vsave`, `vcomponents`, `seeds`, `captions`, `cmethod`, `actionafterfault`, `unsetdummy`, `all`); default `*`
SAVE = *text*	Text to save the contents of the procedure (omitting comments and some spaces)
WORDLENGTH = *string token*	Length of word (32 or 8 characters) to check in identifiers, directives, options, parameters and procedures within the procedure (`long`, `short`); default `*` i.e. no change

Parameter

text	Name of the procedure

Description

Once you start to write programs for complicated tasks, you may wish to keep them to use again in future. The most convenient way of doing this is to form them into procedures. You may also wish to use procedures written by other people. The use of a Genstat procedure looks exactly the same as the use one of the standard Genstat directives. Thus, you simply give the name of the procedure, and then specify options and parameters as required. As with directives, the name can be abbreviated to as few as four characters, provided there is no ambiguity with the names of directives or other procedures. Directives and procedures in the official Genstat library are all defined to have names that are distinct within the first four characters so there should be no problem unless you (or your site) have defined procedures with ambiguous names.

When Genstat meets a statement with a name that it does not recognize as one of the standard Genstat directives, it first looks to see whether you have a procedure of that name already stored in your program. Then it looks in any procedure library that you may have attached explicitly to your program, taking these in order of their channel number (see the `OPEN` directive). The people that manage your computer can define a special *site* library and arrange for this to be attached to Genstat automatically when it is run. If they have done so, this library will be examined next. Finally Genstat looks in the official Genstat Procedure Library, which is also attached automatically to your program. After locating the required procedure, Genstat reads it in, if necessary, and then executes it. So you do not have to do any more than you would to use a Genstat directive.

The official library thus allows new facilities to be offered to all users. Or your computer manager can make procedures available that cover the special needs of the users at your site, and these will over-ride any procedures of the same name in the official library. Or you can form your own libraries of the procedures that you find particularly useful, and these will always be

taken in preference to procedures in the site or the official library. Note however that a procedure cannot have the same name as any of the Genstat directives.

Information is transferred to and from a procedure only by means of its options and parameters. Otherwise the procedure is completely self-contained. Anyone who uses it does not need to know how the program inside operates, what data structures it contains, nor what directives it uses. The data structures inside the procedure are local to the procedure and cannot be accessed from outside.

To write your own procedures, you start by giving a PROCEDURE statement. This has a single parameter which defines the name of the procedure. The name can be up to 32 characters with the same rules as for the identifiers of data structures: the first character must be a letter, the second to the 32nd can be either letters or digits, and characters beyond the 32nd are ignored. However the name cannot be suffixed, and Genstat will warn you if the first four characters are the same as those of a Genstat directive. If so, you will be unable to abbreviate the name fully (down to as few as four characters), but you will need to give enough characters to distinguish it from the directive. If there is ambiguity in the name of a command, Genstat selects the directive or procedure to use according to the following order of priority: directives, user-defined procedures, procedures in libraries attached by the user (in order of channel number), procedures in the site library, and procedures in the official library.

The PARAMETER option indicates whether the settings in any list specified for the parameters of the procedure are to be taken one at a time, or whether they need to be processed together. The difference between these alternatives can be illustrated by considering some of the Genstat directives. For example, with

```
ANOVA Height,Weight; RESIDUALS=Hres,Wres
```

Genstat will first do an analysis with the values in the Height variate and store the resulting residuals in the variate Hres; it then analyses Weight and stores the residuals in Wres. This action corresponds to the default setting PARAMETER=dummy; inside the procedure, each parameter will then be a dummy data structure which will point to each item of the list in turn, in the same way as the parameters of a FOR loop. Conversely, in the statement

```
PRINT Height,Hres
```

the values of Height and Hres are printed together down the page, and this is possible only if PRINT is able to access both variates simultaneously. In a procedure this would require the setting PARAMETER=pointer; each parameter is then a pointer, storing the whole list.

You may change some aspects of the Genstat environment within a procedure. This may be the intended purpose of the procedure; but if it is an unwanted side effect, you should reset them afterwards. The RESTORE option allows you to list aspects that would like Genstat to reset automatically when it finishes executing the procedure. The settings of RESTORE mainly correspond to the various items that can be saved by the ENVIRONMENT and SPECIAL options of the GET directive. The outstyle setting restores the output style of the current output channel (see OPEN and OUTPUT). The setting all which has the same effect as listing all the other settings. Alternatively, you can save and restore the environment explicitly using the GET, SET, ENQUIRE and OUTPUT directives, although this is usually less efficient.

The SAVE option allows you to store the contents of the procedure, up to and including ENDPROCEDURE, in a text so that you can edit and redefine it or, for example, print it to a file or save it on backing store. The saved version is a modified form of the original input. Each line of the text contains a single statement; thus, where a statement spans several lines of input, these are concatenated into a single line in the text (deleting the continuation characters). Any line that contains several statements is split. Comments are removed, and any occurrence of several contiguous spaces is replaced by a single space. Also, a colon is placed at the end of each line.

Finally, the WORDLENGTH option allows you to set the wordlength to be used for identifiers, directives, options, parameters and procedures within the procedure. If WORDLENGTH=long, up

to 32 characters of each of these names are stored and checked; while if `WORDLENGTH=short`, no more than eight characters are used. The default is to keep the existing setting of the wordlength (as in the program defining the procedure).

After the `PROCEDURE` statement, you must define what options and parameters the procedure is to have; this is done by the directives `OPTION` and `PARAMETER` respectively. Only one of each of these should be given, and they must appear immediately after the `PROCEDURE` statement, but it does not matter which of the two you give first. They have very similar syntaxes, except that `OPTION` has an extra parameter which allows you to indicate whether a list of values or of identifiers is allowed. If you do not wish to define options or parameters for a procedure you can simply omit these directives; alternatively you can use `OPTION` or `PARAMETER` but with none of their parameters set, which has precisely the same effect. You can alse use the `CALLS` directive to list any procedures that your procedure calls. (This can be useful when you are defining a suite of procedures that may call each other.)

After the `OPTION`, `PARAMETER` and `CALLS` statements, you then list the statements that are to be executed when the procedure is called: these statements are the sub-program that makes up the procedure. Any data structures defined within the procedure are local to the procedure and cannot be accessed from outside. So you can use any identifiers for the structures, without having to worry about whether they may also be used outside by someone who may later use the procedure. You end these statements making up the procedure by an `ENDPROCEDURE` statement.

You are allowed to redefine an existing procedure if you wish to change any of the statements that it contains. To do this you specify the `PROCEDURE` statement, as usual, followed by the statements making up the new version of the procedure, and then an `ENDPROCEDURE` statement. However, you are not allowed to change the option or parameter definitions, and if there are any changes in the `OPTION` or `PARAMETER` statements, Genstat will give an error diagnostic.

Options: `PARAMETER`, `RESTORE`, `SAVE`, `WORDLENGTH`.
Parameter: unnamed.

See also
Directives: `OPTION`, `PARAMETER`, `CALLS`, `ENDPROCEDURE`, `FAULT`, `COMMANDINFORMATION`, `WORKSPACE`.
Procedure: `CHECKARGUMENT`.
Genstat Reference Manual 1 Summary section on: Program control.

QDIALOG

Produces a modal dialog box to obtain a response from the user.

Options

DIALOG = *string token*	Type of dialog box (checkbox, pushbutton, radiobutton, text, integer, real, variable, query, message); no default, must be specified
TITLE = *text*	Title for the dialog box; default * i.e. none
PREAMBLE = *text*	Informative text that appears above any controls on the dialog; default * i.e. none
LABEL = *text*	Label for the data entry field; default * i.e. none
RESPONSE = *identifier*	Structure to store the response
STATUS = *scalar*	Stores the exit status as 1 for OK, 2 for cancel, 3 for no, or 4 for yes
DEFAULT = *identifier*	Default setting or settings to appear in the menu; default * i.e. none
LIST = *string token*	Whether an interger, real or variable entry field can contain a list of settings (yes, no); default no
HELP = *texts*	Help on the menu, to be displayed in a pop-up window; default * i.e. none
ICON = *string token*	Type of icon to display in the dialog box (information, warning, error, query); default * i.e. none
TIMEOUT = *scalar*	Permits the dialog to continue and return a default value after a specified period (in seconds); default * i.e. no timeout
MINIMUM = *scalar*	Minimum value for numerical input fields; default * i.e. none
MAXIMUM = *scalar*	Minimum value for numerical input fields; default * i.e. none

Parameters

BOXLABEL = *texts*	Label for each checkbox or radio button
BOXRESPONSE = *scalars*	Indicates the selection status of each checkbox or radio button

Description

QDIALOG displays dialog box to obtain a response from the user. The dialog is modal i.e. the Genstat server pauses until an exit button(e.g. OK or Cancel) is pressed on the dialog.

The TITLE option supplies a line of text to appear in the title bar of the dialog box, and the PREAMBLE specified a text of general description to appear at the top of the box. You can use the HELP option to supply additional information, which will be displayed in a pop-up window if the user clicks on a Help button. The ICON option can specify one of the standard windows icons to appear in advance of the preamble. The DEFAULT option can specify a default response to be shown when the dialog appears. The TIMEOUT option can specify a time in seconds, after which the dialog will close automatically, and return the default.

The DIALOG option specifies the type of dialog box, with the following settings.

checkbox	allows the user to make a multiple-choice selection from a set of values. The BOXLABEL parameter specifies textual descriptions for each checkbox. The results selected for the boxes are saved in a list of scalars supplied by the

BOXRESPONSE parameter; each scalar is set to 0 or 1 (false or true) according to whether or not the corresponding box is checked. The DEFAULT option can be set to a scalar or variate containing an index number or numbers defining the check boxes to be checked when the menu first appears. The STATUS option returns the value 1 (for OK) or 2 (for Cancel) depending on how the user exits from the menu.

`radiobutton` is similar to the checkbox dialog but only one choice is permitted. So, if a different choice is selected, the original is automatically cleared. The options and parameters are used in the same way as the checkbox dialog, with two small changes. The DEFAULT option must be set, and the user's choice can be saved (in a scalar) by RESPONSE option, as an alternative to the BOXRESPONSE parameter. The RESPONSE scalar stores the number of the selected button, or zero if the dialog is cancelled.

`pushbutton` works in the same way as a radio button dialog. However, it differs in appearance, as the strings specified in BOXLABEL are displayed on push-buttons. If set, the DEFAULT option specifies the button that is initially selected, and the value that is returned if the user presses the return or space keys.

`text`, `integer` and `real` each contain a field into which a string or number can be typed, as appropriate. The LABEL option can supply a description or prompt for the input field. The input is saved by the RESPONSE option in a text or scalar (or variate if option LIST=yes) as appropriate . The STATUS option returns the value 1 (for OK) or 2 (for Cancel). The MINIMUM and MAXIMUM options can be used to supply constraints for numerical responses.

`variable` contains a field into which identifiers can be typed. The LIST option indicates whether there should be only one identifier, or whether there can be several separated by commas. The LABEL option can supply a description or prompt for the input field. The results are saved by the RESPONSE option, in a pointer. The STATUS option returns the value 1 (for OK) or 2 (for Cancel).

`query` displays the preamble text and icon, followed by the question or query specified by the LABEL option. The dialog has buttons for no and yes. The STATUS option returns the value 1 (for OK), 3 (for no) or 4 (for yes).

`message` displays the preamble text and icon, with an OK button. It exits when the OK button is pressed, with STATUS set to 1 (OK).

Options: DIALOG, TITLE, PREAMBLE, LABEL, RESPONSE, STATUS, DEFAULT, LIST, HELP, ICON, TIMEOUT, MINIMUM, MAXIMUM.
Parameters: BOXLABEL, BOXRESPONSE.

See also

Procedures: QFACTOR, QLIST. QUESTION.

Genstat Reference Manual 1 Summary sections on: Program control, Calculations and manipulation.

QRD

Calculates QR decompositions of matrices.

Option

PRINT = *string tokens*	Printed output required (orthogonalmatrix, uppertriangularmatrix); default * i.e. no printing

Parameters

INMATRIX = *matrices* or *symmetric matrices*	Matrices to be decomposed
ORTHOGONALMATRIX = *matrices*	Orthogonal matrix of each decomposition
UPPERTRIANGULARMATRIX= *matrices*	Upper-triangular matrix of each decomposition

Description

The QR decomposition of a matrix is a decomposition of an m by n matrix A into an orthogonal matrix Q (i.e. $Q'Q = I$), and an n by m matrix R, so that $A = Q\,R$. If m ≥ n, the top n rows of R are triangular and the lower $m-n$ rows contain zeros. If $m < n$, R is trapezoidal, i.e. it has the form $(R_1 \mid R_2)$ where R_1 is an upper triangular matrix and R_2 is a rectangular matrix.

The matrix A to be composed is specified by the INMATRIX parameter, and the matrices Q and R can be saved using the ORTHOGONALMATRIX, and UPPERTRIANGULARMATRIX parameters, respectively.

The PRINT option allows you to print either of the components of the decomposition; by default, nothing is printed.

Option: PRINT.
Parameters: INMATRIX, ORTHOGONALMATRIX, UPPERTRIANGULARMATRIX.

Method

QRD uses subroutines F08AEF and F08AFF from the NAG Library.

See also

Directives: MATRIX, CALCULATE, NAG, FLRV, SVD.
Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Multivariate and cluster analysis.

RANDOMIZE

Randomizes the units of a designed experiment or the elements of a factor or variate.

Options

BLOCKSTRUCTURE = *formula*	Block model according to which the randomization is to be carried out; default * i.e. as a completely-randomized design
EXCLUDE = *factors*	(Block) factors whose levels are not to be randomized
SEED = *scalar*	Seed for the random-number generator; default 0

Parameter

factors or *variates*	Structures whose units are to be randomized according to the defined block model

Description

In its simplest form, RANDOMIZE performs a random permutation of the units of a list of factors or variates. You list these structures with the parameter of RANDOMIZE. Genstat gives them all exactly the same permutation, which is produced by a set of random numbers generated from the SEED option. For example

```
RANDOMIZE [SEED=144556] X,Y
```

puts the values of X and Y into an identical random order. The seed can be any positive integer, but only the last six digits of its integer part are used. Thus the seeds 2144556 and 7144556.3 are both equivalent to the seed 144556. The default of zero continues the existing sequence of random numbers if RANDOMIZE has already been used in the current Genstat job. If RANDOMIZE has not yet been used, Genstat picks a seed at random. On the other hand, if you use the same (non-zero) seed more than once, you will get the same random numbers, and hence the same randomization.

The main use of RANDOMIZE, however, is to randomize the allocation of treatments to units in a designed experiment. In the analysis of designed experiments, the underlying structure of an experiment is defined by the block formula, as explained in the description of the BLOCKSTRUCTURE directive. Provided the only operators in a block formula are the nesting (/) and crossing (*) operators, this also specifies the correct randomization of the experiment.

The nesting operator specifies that one factor is to be randomized within another one. The simplest example is the randomized block design: its block formula is Blocks/Plots; a separate randomization of plots is done for each block. Another example is a split-plot design, the formula for which is Blocks/Wplots/Subplots; this means randomize first the levels of Blocks, then the levels of Wplots within levels of Blocks, and finally the levels of Subplots within the levels of Blocks and Wplots. In other words, there is a separate randomization of Wplots for each Block, and a separate randomization of Subplots for each Wplot. A similar formula and randomization would apply to a resolvable incomplete-block design.

The crossing operator specifies that the factors are to be randomized independently of each other. For example the formula Rows*Cols means randomize the levels of Rows and Cols separately. Thus the same randomization of Cols appears within each Row. This is the block formula associated with a row and column design, for example a Latin square.

You specify the block formula by the BLOCKSTRUCTURE option, which thus defines the way in which the randomization is to be carried out. Genstat does not randomize the factors in the block structure themselves, unless you put them into the parameter list. This is because the original order of the block-factor levels often describes actual positions in the experiment; for example, in a field. So you are most likely to want to keep these values, rather than the random ordering of them that is used to allocate treatments. The block formula for RANDOMIZE must index all the units; so a randomized block block design must be specified for example as

`Blocks/Plots` and not just `Blocks`. To put a formula of just `Blocks` would not give Genstat any information about what to do with the elements of the blocks.

You should use the `EXCLUDE` option if you want to restrict the randomization so that one or more of the factors in the block formula is not randomized. The most common instance where this is required is when one of the treatment factors is time-order, which cannot be randomized.

Options: `BLOCKSTRUCTURE`, `EXCLUDE`, `SEED`.
Parameter: unnamed.

Action with `RESTRICT`
You can randomize only a subset of the units by applying a restriction to any of the vectors in the parameter list. All the vectors will be be treated as though they were restricted and, if more than one is restricted, they must all be restricted in exactly the same way.

See also
Directive: `GENERATE`.
Procedures: `AKEY`, `ARANDOMIZE`, `APERMTEST`, `RPERMTEST`, `SAMPLE`, `SVSAMPLE`.
Functions: `GRBETA`, `GRBINOMIAL`, `GRCHISQUARE`, `GRF`, `GRGAMMA`, `GRHYPERGEOMETRIC`, `GRLOGNORMAL`, `GRNORMAL`, `GRPOISSON`, `GRSAMPLE`, `GRSELECT`, `GRT`, `GRUNIFORM`.
Genstat Reference Manual 1 Summary sections on: Design of experiments, Analysis of variance, Calculations and manipulation.

RBDISPLAY

Displays output from a radial basis function model fitted by RBFIT.

Option

PRINT = *strings*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ

Parameter

pointers	Save structure with details of the fitted model

Description

RBDISPLAY displays results from the fit of a radial basis function model by RBFIT. Details of the fitted model can be supplied using the parameter of RBDISPLAY. This must have been saved using the SAVE parameter of RBFIT. If this is not set, the output is from the most recent model fitted by RBFIT.

Printed output is controlled by the PRINT option, with settings:

description	a description of the model,
estimates	estimates of the parameters,
fittedvalues	fitted values, and
summary	summary (lack of fit etc.).

Option: PRINT.
Parameter: unnamed.

See also

Directives: RBFIT, RBPREDICT.
Genstat Reference Manual 1 Summary section on: Data mining.

RBFIT

Fits a radial basis function model.

Options

PRINT = *string tokens*	Controls fitted output (description, estimates, fittedvalues, summary); default desc, esti, summ
RBTYPE = *string token*	Type of radial basis function (linear, cubic, thinplate, gaussian, multiquadric, inversemultiquadric, cauchy); default line
METRIC = *string token*	How to calculate distances for the radial basis functions (euclidean, cityblock, manhattan, pythagorean); default eucl
SCALING = *string token*	Type of scaling used to compute distances (sd, mahalanobis, supplied); default sd
ALPHA = *scalar*	Specifies the value for the constant α, used to calculate radial distances for RBTPYE settings multiquadric, inversemultiquadric and cauchy; default 1
LAMBDA = *scalar*	Specifies the value of the penalty constant λ
TOLERANCE = *scalar*	Tolerance for setting eigenvalues equal to zero in the singular value decomposition; default 0.000001

Parameters

Y = *variates*	Response variates
X = *pointers*	Independent variates
CENTRES = *pointers*	Centres of the radial basis functions for the dependent variates
RBSCALING = *scalars* or *variates*	Scaling parameters for the radial distance calculations when SCALING=supplied; default 1
FITTEDVALUES = *variates*	Fitted values generated for each y-variate by the model
ESTIMATES = *variates*	Saves the estimated model parameters
EXIT = *scalars*	Saves the exit code
SAVE = *pointers*	Saves details of the model and the estimated parameters for RBDISPLAY or RBPREDICT

Description

RBFIT estimates the parameters of a radial basis function model. The response variate is supplied by the Y parameter, and the independent (or x-) variates are supplied in a pointer by the X parameter.

The model assumes that the y-value on each unit is related to the vector **x** of x-values (x_1 ... x_p) on that unit, according to the model

$$y = \mathrm{f}(\mathbf{x}) + \varepsilon$$

for some unknown function f() and noise ε drawn at random from a Normal distribution with zero mean and unit variance. A radial basis function (RBF) model approximates the function f(·) by a linear combination of t basis functions, giving an approximate fitted value f for the dependent value

$$f = \sum_{k=1\ldots t} w_k h_k + w_{t+1} b$$

where b is a scalar intercept term and h_k is the value given by an RBF for a radial distance z_k between **x** and a centre location $\mathbf{c}_k$ defined for the kth RBF.

The centre locations are supplied in a pointer by the CENTRES parameter. This should have a variate for each x-variate, with a unit for each RBF.

The METRIC option defines how the radial distances are calculated. The default setting,

`euclidean`, uses a scaled Euclidean distance

$$z_k = [(\mathbf{x} - \mathbf{c}_k)\, \mathbf{S}^{-1}\, (\mathbf{x} - \mathbf{c}_k)']^{1/2}$$

where the form of the scaling matrix **S** is controlled by the `SCALING` option (see below). The `cityblock` setting calculates the distance as

$$z_k = \sum_{k=1\ldots t} |x_j - c_{kj}| / s_j$$

where s_j is the jth diagonal element of the scaling matrix **S**. `METRIC` also has settings `pythagorean` and `manhattan` which act as synonyms of `euclidean` and `cityblock`.

The available forms of the scaling matrix, and corresponding settings of the `SCALING` option are as follows:

`sd`	diagonal matrix containing the standard deviations of the x-variates (default),
`mahalanobis`	variance-covariance matrix of the data values of x-variables (to give the Mahalanobis distance),
`supplied`	user-defined scaling parameters, supplied by the `RBSCALING` parameter.

The `mahalanobis` setting is available only for the `euclidean or pythagorean` settings of the `METRIC` option. The setting of `RBSCALING` can be either a scalar or a variate, depending upon the parameters are the same or different over the x-variates; the values must all be greater then zero.

The form φ() of the radial basis functions is specified by the `RNTYPE` option, by selecting one of the following settings:

`linear`	$\varphi(z) = z$,
`cubic`	$\varphi(z) = z^3$,
`thinplate`	$\varphi(z) = z^2 \log_e(z)$,
`gaussian`	$\varphi(z) = \exp(-z^2)$,
`multiquadric`	$\varphi(z) = \sqrt{\{z^2 + \alpha^2\}}$,
`inversemultiquadric`	$\varphi(z) = 1 / \sqrt{\{z^2 + \alpha^2\}}$,
`cauchy`	$\varphi(z) = 1 / (z^2 + \alpha^2)$.

The value of the constant α (which must be positive) is specified by the `ALPHA` option, with a default of one.

The RBF model is fitted by estimating values for the weights w_k. This is done by minimizing the penalized (regularized) sum of squares error function:

$$(\mathbf{y} - \mathbf{f})' (\mathbf{y} - \mathbf{f}) + \lambda \sum_{k=1\ldots t+1} w_k^2$$

where the penalty constant λ must be specified by the `LAMBDA` option.

The inverse-matrix calculations required during the fit are formed using a singular value decomposition. In the calculations, singular values that are less than the largest singular value multiplied by a tolerance are treated as zero. This tolerance is specified by the `TOLERANCE` option; default 0.000001.

Printed output is controlled by the `PRINT` option, with settings:

`description`	a description of the model,
`estimates`	estimates of the parameters,
`fittedvalues`	fitted values,
`summary`	summary (lack of fit etc.).

The `SAVE` parameter can save full detail of the RBF model; this can then be used by the `RBDISPLAY` directive to give further output, or by the `RBPREDICT` directive to form predictions. The estimated weights can be saved using the `ESTIMATES` parameter, and the fitted values vcan be saved by the `FITTEDVALUES` parameter.

Options: `PRINT`, `RBTYPE`, `METRIC`, `SCALING`, `ALPHA`, `LAMBDA`, `TOLERANCE`.
Parameters: `Y`, `X`, `CENTRES`. `RBSCALING`, `FITTEDVALUES`, `ESTIMATES`, `EXIT`, `SAVE`.

Method
RBFIT uses the function nagdmc_rbf from the Numerical Algorithms Group's library of Data Mining Components (DMCs).

Action with RESTRICT
You can restrict the set of units used for the estimation by applying a restriction to the y-variate or any of the x-variates. If several of these are restricted, they must all be restricted to the same set of units.

See also
Directives: RBDISPLAY, RBPREDICT, ASRULES, NNFIT.
Procedures: KNEARESTNEIGHBOURS, RADIALSPLINE.
Genstat Reference Manual 1 Summary section on: Data mining.

RBPREDICT

Forms predictions from a radial basis function model fitted by RBFIT.

Option

PRINT = *strings*	Controls fitted output (description, predictions); default desc, pred

Parameters

X = *pointers*	X-values at which to predict
PREDICTIONS = *variates*	Predictions
SAVE = *pointers*	Details of the fitted model

Description

RBPREDICT forms predictions using radial basis function model fitted by RBFIT. Details of the the model and the estimated parameters are supplied using the SAVE parameter. This must have been saved using the SAVE parameter of RBFIT. If this is not set, the output is from the most recent model fitted by RBFIT. The values of the x-variates at which to predict are supplied, in a pointer, using the X parameter. The variates in the pointer must be in exactly the same order as the equivalent variates in the pointer defined for the X parameter in the original RBFIT command.

The output is controlled by the PRINT option, with settings:

description	a description of the model,
predictions	predicted values.

Option: PRINT.
Parameters: X, PREDICTIONS, SAVE.

Method

RBPREDICT uses the function nagdmc_predict_RBF from the Numerical Algorithms Group's library of Data Mining Components (DMCs).

Action with RESTRICT

You can restrict the set of units used for the prediction by applying a restriction to any of the x-variates. If several of these are restricted, they must all be restricted to the same set of units.

See also

Directives: RBDISPLAY, RBFIT.
Genstat Reference Manual 1 Summary section on: Data mining.

RCYCLE

Controls iterative fitting of generalized linear, generalized additive and nonlinear models, and specifies parameters, bounds etc for nonlinear models.

Options

MAXCYCLE = *scalars*	Maximum number of iterations for Fisher-scoring algorithm (used in generalized linear models), back-fitting algorithm (used in additive models) and nonlinear algorithms; single setting implies the same limit for all; default 15, 15, 30
TOLERANCE = *scalar* or *variate*	Scalar or first unit of a variate defines the convergence criterion for the relative change in deviance and, if required, the second element of a variate defines the criterion for convergence to a zero deviance; default !(0.0001,1.0E-11)
FITTEDVALUES = *variate*	Initial fitted values for generalized linear model; default *
METHOD = *string token*	Algorithm for fitting nonlinear model (GaussNewton, NewtonRaphson, FletcherPowell); default Gaus, but Newt for scalar minimization
LINEARPARAMETERS = *scalars*	Scalars to hold current values of linear parameters used in nonlinear model, for reference within model calculations

Parameters

PARAMETER = *scalars*	Nonlinear parameters in the model
LOWER = *scalars*	Lower bound for each parameter
UPPER = *scalars*	Upper bound for each parameter
STEPLENGTH = *scalars*	Initial step length for each parameter
INITIAL = *scalars*	Initial value for each parameter

Description

RCYCLE can be used, after MODEL, to modify aspects of the optimization process used by later FIT, FITCURVE and FITNONLINEAR directives.

The MAXCYCLE option can be set to a list of three scalars to specify respectively the maximum number of iterations to be used in the Fisher-scoring algorithm used to fit a generalized linear model, the back-fitting algorithm used in generalized additive models, and the algorithms for nonlinear models. These have the defaults 15, 15 and 30. If a single value is supplied, it is taken to apply to all three situations.

The TOLERANCE option can be set to a scalar or a variate to control the criterion for convergence in generalized linear and generalized additive models. A scalar or the first unit of a variate defines the convergence criterion for the relative change in deviance (default 0.0001). The iteration stops when the absolute change in deviance in successive cycles is less than the tolerance multiplied by the current value of the deviance. The second element of a variate defines the criterion for convergence to a zero deviance. If TOLERANCE is unset, or if it is set to a scalar, the default criterion for zero deviance is 1.0E−11.

The algorithm for generalized linear models has to start by estimating an initial set of fitted values. Genstat usually obtains these by a simple transformation of the observed responses. It may be that better estimates are available, for example from a previously fitted model; if so, you can supply them by the FITTEDVALUES option.

The PARAMETER, STEPLENGTH and INITIAL parameters can be used to supply initial step lengths and initial values for the nonlinear parameters in the standard curves fitted by FITCURVE,

although this will usually not be necessary; FITCURVE has effective ways of its own to ascertain good starting value for each parameter, for example by a short grid search or by some manipulation of the data values. The parameters must be listed in the same order as Genstat uses to print them. RCYCLE defines the identifiers as scalars holding the initial values that you have supplied; after the model has been fitted they contain the estimated values of the parameters.

The other parameters are relevant only to general nonlinear models fitted by FITNONLINEAR or FIT. The PARAMETER parameter then merely lists the scalars that are used to represent the nonlinear parameters in the model calculations, the LOWER and UPPER parameters specify bounds, the STEPLENGTH parameter specifies initial step lengths, and the INITIAL parameter specifies initial values. The METHOD option is also relevant only for general nonlinear models, when it specifies the optimization method to be used. (Bounds are determined automatically for standard curves, and Genstat then always uses a modified Newton method.)

Options: MAXCYCLE, TOLERANCE, FITTEDVALUES, METHOD, LINEARPARAMETERS.
Parameters: PARAMETER, LOWER, UPPER, STEPLENGTH, INITIAL.

See also
Directives: FIT, FITCURVE, FITNONLINEAR.
Genstat Reference Manual 1 Summary section on: Regression analysis.

RDISPLAY

Displays the fit of a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, confidence); default mode,summ,esti
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
DISPERSION = *scalar*	Dispersion parameter to be used as estimate for variability in s.e.s; default is as set in the MODEL statement
RMETHOD = *string token*	Type of residuals to display (deviance, Pearson, simple); default is as set in the MODEL statement
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default is as set in the MODEL statement
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
DFDISPERSION = *scalar*	Allows you to specify the number of degrees of freedom for a dispersion parameter specified by the DISPERSION option; default is as set in the MODEL statement
SAVE = *identifier*	Specifies save structure of model to display; default * i.e. that from latest model fitted

Options

No parameters

Description

RDISPLAY produces further output from a linear, generalized linear, generalized additive or nonlinear model. The PRINT option has the same settings as in the FIT directive, except that no

monitoring is available. The CHANNEL option selects the output channel to which the results are output, as in the PRINT directive; this may be a text structure, allowing output to be stored prior to display. The DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION and PROBABILITY options are also as in the FIT directive.

The RMETHOD option allows you temporarily to change the method of forming residuals, for the output of the current statement only, in the same way as the corresponding option in the MODEL directive sets the default method of formation. Similarly, the DMETHOD option temporarily changes the method used to calculate the residual variability to be displayed for a generalized linear model, the DISPERSION option allows you (temporarily) to set the dispersion parameter, and the DFDISPERSION option allows you to define the number of degrees of freedom for a specified dispersion parameter. These again operate like the corresponding options of MODEL (except that they apply only to the current statement).

The SAVE option lets you specify the identifier of a regression save structure; the output will then relate to the most recent regression model fitted with that structure.

Options: PRINT, CHANNEL, DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION, DISPERSION, RMETHOD, DMETHOD, PROBABILITY, DFDISPERSION, SAVE.
Parameters: none.

See also
Directives: MODEL, FIT, FITCURVE, FITNONLINEAR, PREDICT.
Procedures: RCHECK, RGRAPH, RDESTIMATES, RCOMPARISONS, RTCOMPARISONS, RWALD, FIELLER, RFUNCTION.
Genstat Reference Manual 1 Summary section on: Regression analysis.

READ

Reads data from an input file, an unformatted file or a text.

Options

PRINT = *string tokens*	What to print (data, errors, summary); default erro, summ
CHANNEL = *identifier*	Channel number of file, or text structure from which to read data; default current file
SERIAL = *string token*	Whether structures are in serial order, i.e. all values of the first structure, then all of the second, and so on (yes, no); default no, i.e. values in parallel
SETNVALUES = *string token*	Whether to set number of values of vectors from the number of values read (yes, no); default no causes the number of values to be set only for structures whose lengths are not defined already (e.g. by declaration or by UNITS)
LAYOUT = *string token*	How values are presented (separated, fixedfield); default sepa
END = *text*	What string terminates data (* means there is no terminator); default ':'
SEQUENTIAL = *scalar*	To store the number of units read (negative if terminator is met); default *
ADD = *string token*	Whether to add values to existing values (yes, no); default no (available only in serial read)
MISSING = *text*	What character represents missing values; default '*'
SKIP = *scalar*	Number of characters (LAYOUT=fixe) or values (LAYOUT=sepa) to be skipped between units (* means skip to next record); default 0 (available only in parallel read)
BLANK = *string token*	Interpretation of blank fields with LAYOUT=fixe (missing, zero, error); default miss
JUSTIFIED = *string tokens*	How values are to be assumed justified with LAYOUT=fixe (left, right); default righ
ERRORS = *scalar*	How many errors to allow in the data before reporting a fault rather than a warning, a negative setting, -*n*, causes reading of data to stop after the *n*th error; default 0
FORMAT = *variate*	Allows a format to be specified for situations where the layout varies for different units, option SKIP and parameters FIELDWIDTH and SKIP are then ignored (in the variate: 0 switches to fixed format; 0.1, 0.2, 0.3 or 0.4 to free format with space, comma, colon or semi-colon respectively as separators; * skips to the beginning of the next line; in fixed format, a positive integer *n* indicates an item in a field width of *n*, −*n* skips *n* characters; in free format, *n* indicates *n* items, −*n* skips *n* items); default *
QUIT = *scalar*	Channel number of file to return to after a fatal error; default * i.e. current input file
UNFORMATTED = *string token*	Whether file is unformatted (yes, no); default no
REWIND = *string token*	Whether to rewind the file before reading (yes, no); default no

SEPARATOR = *text*	Text containing the (single) character to be used in free format; default `' '`
SETLEVELS = *string token*	Whether to define factor levels or labels (according to the setting of FREPRESENTATION) automatically from those that occur in the data (yes, no); default no causes them to be set only when they are not defined already
TRUNCATE = *string tokens*	Truncation of leading or trailing spaces of strings read in fixed format (leading, trailing); default * i.e. none
CASE = *string token*	Whether the case of letters (small and capital) should be regarded as significant or ignored when forming factor labels automatically (significant, ignored); default sign
LDIRECTION = *string token*	How to define the ordering of levels or labels when these are formed automatically (ascending, given); default asce

Parameters

STRUCTURE = *identifiers*	Structures into which to read the data
FIELDWIDTH = *scalars*	Field width from which to read values of each structure (LAYOUT=fixe only)
DECIMALS = *scalars*	Number of decimal places for numerical data containing no decimal points
SKIP = *scalars*	Number of values (LAYOUT=sepa) or characters (LAYOUT=fixe) to skip before reading a value
FREPRESENTATION = *string tokens*	How factor values are represented (labels, levels, ordinals); default leve

Description

Data values can be read into any Genstat data structure using the READ directive. In its simplest form, you merely list the structure whose values are to be read: for example

```
READ Weight
```

The data values for Weight are then assumed to come on the following line or lines. They are assumed to be in *free format*, separated one from another by one or more spaces or tabs or new lines, and to be terminated by a colon.

READ has a PRINT option with settings:

summary	to print a summary of the data
data	to print a copy of the input lines
errors	to print a detailed report on any errors in the data

By default PRINT=summary,errors.

The CHANNEL option allows you to read data from another file; this must already have been opened (see the OPEN directive). You can also read data from a Genstat text structure. Each line of input is then treated as if it had been read from a file. Note: you should use CHANNEL if you want to use READ in an IF or CASE structure, a FOR loop or a procedure.

You can read values for more than one structure in a single READ statement. The values can be taken either *serially* or in *parallel*. The default is to take the values in parallel: the first element of each structure is read, then the second element of each, until all the data are read. For example:

```
a1 b1 c1
a2 b2 c2
a3 b3 c3
a4 b4 c4 :
```

or

```
a1 b1 c1 a2
b2 c2
a3 b3 c3 a4 b4 c4 :
```

Here A, B and C are in parallel, each with four values. The complete set of values for all three structures is given, followed by one terminating colon. The term *parallel* merely indicates the order in which READ is to read the values: that is, the first element of each structure, then the second element of each, and so on. It is not necessary for the data to be laid out in neat columns, although this may make a data file easier to work with. Different types of structures can be read in parallel and they may have different kinds of values (numerical or text).

Alternatively, you can set option SERIAL=yes to read the structures in *series*. Then all the values of the first structure are read, followed by all the values for the second structure, and so on, until all the data structures have been read. For example

```
x1 x2 x3 :
y1 y2 :
z1 z2 z3 z4 z5 z6 :
```

Here all the values of X are given first, followed by all the values for Y, and then all the values for Z. Unlike the parallel layout, each set of values must end with the terminating colon, so that READ can tell when to move on to the next structure; this means that the structures can be of different lengths.

When you are working interactively, Genstat produces a prompt indicating the name of the data structure and the unit number of the next value it expects to read. If Genstat knows how many values to expect, it will terminate the input automatically, without asking for the terminating colon, if the last value is at the end of a line. However, it is quite correct to include the colon at the end of that line of data if you want. If you type too many values by mistake you will get a warning message telling you that the extra data has been ignored.

If a structure whose values are to be read has not already been declared, Genstat will define it automatically as a variate. Likewise, if the length of a vector is undefined, this too will be set automatically. READ first checks whether the vector is being read in parallel with other vectors whose lengths have been defined, then it looks to see if a default length has been defined for vectors using the UNITS directive. If neither of these is available to define the length, it is set to the number of data values that are provided in the input. Lengths of vectors can also be *redefined* according to the number of data values that are read, by setting option SETNVALUES=yes. The END option allows you to define another string of characters to be used instead of a colon to mark the end of the data, or you can set END=* to indicate that there is no terminating string.

The values of numerical structures (scalars, variates, matrices, symmetric and diagonal matrices and tables) can be entered in any of the standard forms: for example

```
1.20  -.2   3e1  -1.25E-2  27
```

are all valid.

Textual values (strings) in free format must be enclosed within single quotes if they contain any characters that have special meaning to READ (space, tab, comma, colon, asterisk, backslash, single or double quote). The quotes can be omitted for other strings. For example:

```
TEXT [NVALUES=5] Country
READ Country
Australia   Canada  'Great Britain'  U.S.A.  'New Zealand' :
```

The rules for strings in READ are thus slightly different to those for lists of strings, where quotes are required for any string that does not start with a letter or contains any character other than letters or digits. Thus Newcastle-on-Tyne and 500Km are both valid when read in as data, but not in a TEXT declaration. Rules for strings in fixed format are described later.

The values of factors are usually represented by their levels. You can change this by setting the FREPRESENTATION parameter. If you set it to labels, READ will accept as values the labels

of the factor, using the same rules as for reading textual strings. The strings given as data values must match exactly the labels of the factor if they have been declared. The setting FREPRESENTATION=ordinals causes READ to expect an integer in the range 1 up to *n*, the number of levels declared for the factor. As FREPRESENTATION is a parameter it can be set to a list of values which are cycled in parallel with the structures to be read. Thus, you are allowed to read several factors in one READ statement, possibly using a different method for reading each one. The setting of this parameter is ignored for any structures that are not factors, but remember that the list will still be cycled in parallel with these other structures.

If you set option SETLEVELS=yes, READ will set up the factor levels or labels according to the values that it finds when reading the data. By default it distinguishes between capital and small letters when forming factor labels, but you can set option CASE=ignored to ignore the case of letters. Also, by default the levels or labels are sorted into ascending order, but you can set option LDIRECTION=given to leave them in the order in which they are found in the data file.

The values of pointers are identifiers, that is, names of other data structures. When reading a pointer only simple identifiers are allowed: suffixes cannot be used. For example, Winston is allowed but Orwell[1984] is not.

You cannot read formulae or expressions directly. The easiest way to do this is to read the required value into a text which can then be used in an appropriate declaration using either the macro-substitution symbols ## or the EXECUTE directive. You cannot read values into compound data structures; these should be formed using the appropriate directives or by reading their components individually.

By default, a missing value should be indicated by an asterisk (*); this means that any data item that begins with * is treated as missing. For example, any of the three strings

```
*   ***   *789
```

will be treated as missing. You can use the MISSING option to change this to any other single character; for example, if you set MISSING='-' then any negative numbers will be read as missing values.

In free format, values are usually separated by spaces or tabs. The SEPARATOR option can be used to specify another character to use as a separator. For example you can use a comma:

```
READ [SEPARATOR=','] Weights
24.3, 25.6, 57.3, 43.8, 45.3,
46.5, 47.9, 97.0, 77.5, 64.3 :
```

You can use spaces and tabs in addition to the specified separator, so long as the separator is present between each pair of values (except at the end of line, when it may be omitted).

The SEPARATOR, END and MISSING strings are all case-sensitive; for example, END=enddata is different from END=EndData. The missing-value and separator characters must be distinct and neither may be part of the END string.

In free format, the SKIP option can be used to skip values between complete units of data. For example, with a file in channel 2 containing five columns of data, the statement

```
READ [CHANNEL=2; SKIP=3] X,Y
```

would read X and Y from the first two columns, and then skip the final three columns: Genstat reads the first value for X and Y, the next three values are skipped before reading the second value of X; so READ moves onto the next line of the file, and so on. You can also set SKIP=* to skip directly to the next line of data; you could use this if there were varying numbers of additional columns in the file. By default, SKIP is zero, so no values are skipped. The SKIP parameter is interpreted in parallel with the structures whose values are to be read, and indicates how many values should be skipped before reading the value for the corresponding structure.

In fixed format, data values are arranged in specific *fields* on each line of the file. Each field consists of a fixed number of characters. There is no need for separating spaces; the tab character is not permitted, nor are comments. So, depending on how the fields are defined, the sequence

of digits `123456` could be interpreted for example as the single number 123456, or two numbers 123 and 456, or three numbers 123, 4 and 56. Data like this are usually produced by special-purpose programs or equipment; for example, automatic data recorders.

To read data in fixed format you set the `LAYOUT` option to `fixed`, and then specify the format to be used. If the values for a structure always occupy the same number of character positions, you can do this with the `FIELDWIDTH` parameter. For example,

```
READ [CHANNEL=2; LAYOUT=fixed] Weight,Height; FIELDWIDTH=3,5
```

takes data from channel 2 in fixed format. The data are in parallel: that is, reading across lines of the file, values for `Weight` and `Height` appear alternately. The `FIELDWIDTH` parameter is processed in parallel with the structures to be read, so each item of `Weight` data takes up three characters, and each item of `Height` data takes up five. If the fieldwidth for a structure is not constant, that is if different layouts are used for different units of the data, then you need to use the `FORMAT` option, described later.

Suppose there are 80 characters per line in the file; each pair of `Weight` and `Height` values takes up 8, and so you have 10 pairs per line. The first line looks like:

$\texttt{Weight}_1\texttt{Height}_1\texttt{Weight}_2\texttt{Height}_2$ ... $\texttt{Weight}_{10}\texttt{Height}_{10}$

Suppose that the first two values for Weight were 1 and 200, and that the first two for Height were 10 and 1200. Then, using ␣ to represent a space, the first four items on this line would be:

␣␣1␣␣␣10200␣1200

Genstat is able to identify the separate values 10 and 200 because it is reading a fixed number of characters for each structure.

Genstat input files have a nominal width, set by default to 80. This can be altered by an `OPEN` statement to a different value if necessary. When reading in fixed format, each line of input is taken to be exactly this width; shorter lines are extended with spaces (blanks). It is important to make sure that you account for this when setting the options for `READ`, otherwise you may read some values from these blank fields (the `BLANK` option, described below, explains how the blank fields would be interpreted). In the example above, if the values for `Height` occupied four characters instead of five there would be 11 pairs of values per line of 77 characters. Using the default settings, the final three characters on the first line would be read as the 12th value of `Weight`, and `READ` would then be out of step as the 12th value of `Height` would be read in from the beginning of the next line. The simplest solution is to set the file width to 77 in the `OPEN` statement, but you can also use the `SKIP` option and parameter (see below) or the `FORMAT` option to avoid this sort of problem.

When you are using fixed format, the data terminator must begin within the first field to be read after the final data value: so you must ensure that you set the field widths and position the terminator appropriately. If you are using either the `SKIP` option or parameter, you must take care not to skip accidentally over the terminator, as `READ` will continue to take input - and probably generate many error messages.

Normally Genstat treats a blank field in fixed-format data as a missing value, and the only indication will be in the count of missing values in the printed summary. You can request warning messages for blank fields by setting the option `BLANK=error`. Alternatively, you can cause blanks to be interpreted as zeroes, by setting `BLANK=zero`.

Data in fixed format are normally taken to be right-justified: that is, their right-hand ends are flush with the right-hand end of the field; you can have either blanks or leading zeroes (for numbers) in the redundant spaces at the left of the field. You can change this default by setting the `JUSTIFIED` option. For example the value 123 can appear in a field of width 5 as:

␣␣123	`JUSTIFIED=right`	there may be leading blanks (the default),
123␣␣	`JUSTIFIED=left`	there may be trailing blanks,
00123	`JUSTIFIED=left,right`	there must be no blanks, or

```
␣123␣  JUSTIFIED=*        there may be leading or trailing blanks.
```

In this way, JUSTIFIED allows you to check the blanks in each field. If a data field contains any blanks that are not allowed by the current setting, an error will be reported. Note that when reading numerical data embedded blanks are never permitted. So a field containing, for example 1␣2␣3, will always produce an error message.

As an example, we can read the values of five scalars using a fixed format with values left-justified in their fields by the following:

```
SCALAR V,W,X,Y,Z
READ [LAYOUT=fixed;JUSTIFIED=left] V,W,X,Y,Z; \
                          FIELDWIDTH=4,5,7,4,5
1.235.62␣678.9␣␣3.7810.31:
```

This reads the values 1.23, 5.62, 678.9, 3.78 and 10.31 into V, W, X, Y and Z respectively.

The general principles of the SKIP option and parameter are discussed in the context of a free format read in the previous section. When reading in fixed format the same ideas apply, but the SKIP settings now specify numbers of characters to be ignored, instead of numbers of values. Thus, you can obtain exactly the same effect as in the example above by putting

```
READ [LAYOUT=fixed] V,W,X,Y,Z; FIELDWIDTH=4,4,5,4,5; \
                    SKIP=0,0,1,2,0
```

Sometimes fixed format data can be further compressed by omitting the decimal point. The DECIMALS parameter allows you to re-scale data automatically when it is read (in either fixed of free format).

When reading textual data in fixed format, the contents of each field are taken exactly as they appear in the input file. There is no need to enclose values in quotes; in fact if you do so, the quotes are treated as part of the data. For example,

```
TEXT [NVALUES=1] T1,T2,T3,T4
READ [LAYOUT=fixed; SKIP=*] T1,T2,T3,T4; FIELDWIDTH=6,3,4,7
'What's␣it␣all␣about?':
```

gives text T1 the value 'What's, text T2 the value ␣it, text T3 the value ␣all, and text T4 the value ␣about?'. Consequently, the only way to represent a missing string in fixed format is by a blank field, as '' or * would both be treated literally and stored as data values.

The TRUNCATE option has settings leading and trailing, allowing you to remove initial or trailing spaces in strings that are read in fixed format. For example, if we set TRUNCATE=leading above, T2 would just contain the two letters it. By default no truncation takes place.

The rules for reading textual data in fixed format also affect the reading of factors. If you set FREPRESENTATION=labels and do not request any truncation, the width of the field must equal the number of characters in the label, as for example no␣ is not the same as no.

The FORMAT option allows you to use use a *variable format*. By this we mean that the layout of the values may vary from unit to unit of the data, and may also vary within each unit. For example, suppose you have some meteorological data which was measured daily and that the file also contains some additional summary values at the end of each week. The first eleven lines are reproduced to illustrate the structure of the file:

```
Monday          5.5      -0.4       0.0       1.9       10.0
Tuesday        -1.1      -2.1       0.0       0.0       34.0
Wednesday       0.6      -8.3       1.3       5.4      142.0
Thursday        6.8      -5.7       1.1       0.0      158.0
Friday         10.6       0.5       8.1       0.0      141.0
Saturday       10.7       6.4       8.3       0.0      152.0
Sunday         10.0       1.9       1.0       0.1      237.0
Summary week 1>  10.7  -8.3  4  19.8  7.4  10.0  124.8  237.0
Monday          9.9       2.5       0.0       4.4      229.0
Tuesday        11.4       2.1       8.5       0.3      237.0
Wednesday      11.9       6.3      18.7       0.0      520.0
```

Suppose the file contains data for 28 days. If you try to read a text and five variates of length 28 then the summaries found after the 7th, 14th, 21st and 28th days would cause an error in READ. You need to read seven lines, skip one, read seven more, and so on. This can be done by setting the option FORMAT=!((6)7,*,*). This means "read six values, do this seven times, skip to the next line, skip again, then return to the beginning of the format and repeat, until enough data has been read". The format is made clear by using (6)7 which corresponds to the physical layout of the data, but 42 could have been specified instead, meaning read the next 42 values.

You can use FORMAT when reading in either free format or fixed format, and can also switch between the two during the READ. When you have set FORMAT, Genstat ignores the SKIP option and the FIELDWIDTH and SKIP parameters, and READ is controlled entirely by the values of the FORMAT. These values are not in parallel with the list of structures: they apply to data values in turn, recycling from the beginning when necessary. You set FORMAT to a variate, which may be declared in advance or can be an unnamed structure as shown above. Each value of this variate is interpreted as follows (where *n* is a positive integer):

+*n* read *n* values (in free format) or one value from a field of *n* characters (in fixed format);
-*n* skip the next *n* values (in free format) or *n* characters (in fixed format)
* skip to the beginning of the next line
0.0 switch to fixed format
0.1 switch to free format using space as a separator
0.2 switch to free format using comma as a separator
0.3 switch to free format using colon as a separator
0.4 switch to free format using semicolon as a separator
0.5 switch to free format using the setting of the SEPARATOR option

Using the FORMAT variate READ will start in either free format or fixed format, according to the setting of LAYOUT (by default, LAYOUT=separated; that is, free format). You can switch between these at any time by specifying a value in the range 0-0.5. Remember that if you use free format, spaces and tabs can also be used in addition to the specified separator, and you must use a separator that is distinct from the END and MISSING indicators.

You can read from unformatted files by setting option UNFORMATTED=yes. The only options that are then relevant are CHANNEL, REWIND and SERIAL. Details of how to create the unformatted files are given in the description of the PRINT directive.

If you have more data to read than can be stored in the space available within Genstat, you can use the SEQUENTIAL option of READ to process the data in smaller batches. This works by reading in some of the data, partially processing it to form an intermediate result, and then overwriting the original data with a new batch that is used to update the intermediate results. This can be repeated until all the data has been read and the final summary is obtained. There are two directives that include facilities specifically designed to work with sequential data input: TABULATE which forms tabular summaries, and FSSPM which forms SSPM data structures for use in linear regression. You can also use other directives, such as CALCULATE, to process data sequentially, but you will have to program the sequential aspects yourself.

You should first declare the structures to be of some convenient size, such that you will not use up all the work space. You then use READ as normal, but with the SEQUENTIAL option set to the identifier of a scalar, which will be used to keep track of how the input is progressing. For example, to read in 10 variates of length 272500:

```
VARIATE [NVALUES=10000] X[1...10]
READ [CHANNEL=2; SEQUENTIAL=N] [1...10]
```

The number of values declared for X[1...10] defines the size of batch to read (10000 in this example). So, READ will read the first 10000 units of data (100,000 values), and set N to 10000 to indicate that is the number of units read. This should be followed by the statements to process the first batch of data, then the READ can be repeated. Once again N is set to 10000, indicating that another 10000 units have been read. This can be continued until READ finds the data

terminator, when it sets the sequential indicator to minus the number of values found in the last batch. If this is less than the declared size of the data structures they will be filled out with missing values. In the example given above, after the 28th READ the variates will each contain 2500 values followed by 7500 missing values, and N will be set to −2500, indicating that all the data has been read and that the final batch contains only 2500 values. Usually you will use the SEQUENTIAL facility in conjunction with FSSPM or TABULATE which are designed to recognize the different settings of the scalar N.

The SEQUENTIAL option is best used within a FOR loop. You should set the NTIMES option to a value large enough to ensure that sufficient batches of data are read. The loop should contain the READ statement and any other statements required to process the data. For example

```
VARIATE [NVALUES=10000] X[1...10]
SSPM [TERMS=X[]] S
FOR [NTIMES=9999]
  READ [PRINT=*;CHANNEL=2;SEQUENTIAL=N] X[]
  FSSPM [SEQUENTIAL=N] S
  EXIT N.LE.0
ENDFOR
```

The EXIT directive is used to jump out of the loop once all the data has been read and processed; this is safer than trying to program an exact number of iterations for the loop. The exit condition includes the case when N is equal to zero, as this will arise when the batch size exactly divides the total number of units. In the above example, if there were 280000 units of data altogether, the 28th READ would terminate with N set to 10000. This is because READ is unable to look ahead for the terminator, as there may be other statements in the loop, such as SKIP, which affect how the file is read. The next READ would immediately find the data terminator, so would exit with N set to zero. This special case is treated appropriately by FSSPM and TABULATE, but you should remember to allow for it if you are programming the sequential processing explicitly.

You can use the SEQUENTIAL option to read data from more than one input channel, perhaps when a large data set is split into two or more files, but you are not allowed to read data from the current input channel (that is, the channel containing the READ statement). If you want to process several structures sequentially from the same file, you must read them in parallel. You must also be careful not to modify the value of the scalar, N, within the loop when using sequential data input with FSSPM or TABULATE, as that could interfere with the sequential processing.

Another means of handling large amounts of data is provided by the ADD option. This allows you to add values to those already stored in a structure, thus forming cumulative totals without having to store all the individual data values. You must set SERIAL=yes with ADD=yes; and it is allowed only for variates. For example:

```
VARIATE [NVALUES=6] A
READ [ADD=yes; SERIAL=yes] 3(A)
5 12 9 * * 9 :
8 1 3 * 2 10 :
3 4 0 * 11 * :
```

This starts by assigning the values 5, 12, 9, *, *, and 9 to A. Then A is read again, and its values become 13, 13, 12, *, 2, 19: with ADD=yes (and only then) missing values are interpreted as zeroes when being added to non-missing values. Finally A contains the values 16, 17, 12, *, 13, 19.

If you have used the UNITS directive to specify a variate or text containing unit labels, READ will respect the order of these values when reading other structures in parallel with the units structure; in other words the data are re-ordered to match the order of the unit labels. If the units structure does not already have values, READ will define order of the units as the order in which it finds them in the data. This means that if you are reading several sets of data, each having a column for the unit number (or label), the first use of READ will define the unit order and subsequent READ statements will ensure that this order is maintained consistently in the

remaining data. If a value is specified more than once when defining the units structure, READ will only ever locate the first occurrence of that unit label. If a unit label is repeated in the data then only the final set of values corresponding to that unit will be stored; earlier occurrences are overwritten by subsequent ones. If you try to read a value that is not present in the units structure this is regarded as a fault. Also, if the units structure contains missing values it cannot be used to re-order the data and will instead be overwritten by the new values: a warning message is printed out to tell you if this occurs. If you use the option SETNVALUES=yes when reading structures in parallel with the units vector, the other structures will all be set to the current unit length.

When you are working interactively and typing data from the keyboard, READ will halt immediately it finds an invalid value. You should type the correct value and then continue with the rest of the data. If you had typed several items of data then all those before the erroneous value will have been read and stored, but any remaining values will have been discarded, and so will need to be retyped. When you are reading data in batch, it is not possible to recover from errors in this way. Instead, READ will continue processing the data, substituting missing values for any data that it cannot read, and printing out a message for every error that is found.

If errors occur when running in batch, a fault will be generated when READ terminates, thus terminating the job. This is to avoid spurious output being produced from analyses based on incorrect data. You can override this by using the options ERRORS and QUIT. If you set ERRORS=*n*, where *n* is a positive integer, then up to *n* errors are allowed in the data before READ generates a fault. You might want to do this if you knew certain items of data were going to generate errors, but were prepared to accept them as missing values so that you could analyse the rest of the data. Obviously, you need to be very careful when doing this, as there may be other unexpected errors in the data. Usually you would have to try reading the data once without setting ERRORS, so you could check all the messages, and find what value of *n* is appropriate. Then the READ statement would have to be repeated, setting ERRORS and REWIND in order to read the data. For example, if missing values of a factor had been typed in as the letter X, you would not want to define X as an extra level of the factor, but if you set MISSING='X' any numerical data that used * for missing value could not be read either.

READ produces a message for every data value that contains an error. This can be very useful, as you then have the opportunity to correct all the errors at once, before trying to read the data again. However, the error messages may not be due to errors in the data, but may be caused by an incorrectly specified READ statement. For example, if you are reading many structures in parallel and specify texts and variates in the wrong order in the list of structures to be read, you will get an error message every time Genstat finds a piece of text rather than a number in the position specified for a variate. This is not likely to be a problem, unless you are reading large amounts of data, when you might end up with thousands of lines of needless error messages. A sensible precaution then is to request Genstat to abort the READ if more than a specified number of errors occur. You can do this by setting ERRORS to a negative integer, $-n$. This means that up to *n* errors are allowed in the data, but READ will abort if any more occur, switching control to the channel specified by QUIT (that is, starting or continuing to read Genstat statements from that channel). If you are working in batch a fault will be generated that inhibits execution of further statements, but interactively you have the opportunity to examine the data that have been read in so far, which may help identify any problems in the original READ statement or declarations of your data.

Options: PRINT, CHANNEL, SERIAL, SETNVALUES, LAYOUT, END, SEQUENTIAL, ADD, MISSING, SKIP, BLANK, JUSTIFIED, ERRORS, FORMAT, QUIT, UNFORMATTED, REWIND, SEPARATOR, SETLEVELS, TRUNCATE, CASE, LDIRECTION.

Parameters: STRUCTURE, FIELDWIDTH, DECIMALS, SKIP, FREPRESENTATION.

Action with `RESTRICT`
`READ` ignores any restrictions.

See also
Directives: `OPEN`, `COPY`, `RETRIEVE`, `SKIP`, `SPLOAD`.
Procedures: `FILEREAD`, `IMPORT`, `DBIMPORT`, `TX2VARIATE`.
Genstat Reference Manual 1 Summary section on: Input and output.

RECORD

Dumps a job so that it can later be restarted by a RESUME statement.

Option

CHANNEL = *scalar*	Channel number of the backing-store file where information is to be dumped; default 1

No parameters

Description

The RECORD directive sends all the relevant information about the current state of your Genstat job to the backing-store file specified by the CHANNEL option. You can then use the RESUME directive, either later in your program, or during a completely different Genstat run, to recover all this information and continue your use of Genstat from that point. This can be useful if you need to abandon an analysis and resume it at some later date, or if you want to save the current state of a program in case your next operations turn out to be unsuccessful. The information includes the attributes and values of all your data structures, procedures, and the current graphics settings, but no details are kept of the files that are open on any of the channels. If you use RECORD with the same channel number again, the earlier information is overwritten.

Option: CHANNEL.
Parameters: none.

See also

Directives: RESUME, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

REDUCE

Forms a reduced similarity matrix referring to the GROUPS instead of the original units (synonym of HREDUCE).

Options

PRINT = *string token*	Printed output required (similarities); default * i.e. no printing
METHOD = *string token*	Method used to form the reduced similarity matrix (first, last, mean, minimum, maximum, zigzag); default firs

Parameters

SIMILARITY = *symmetric matrices*	Input similarity matrix
REDUCEDSIMILARITY = *symmetric matrices*	Output (reduced) similarity matrix
GROUPS = *factors*	Factor defining the groups
PERMUTATION = *variates*	Permutation order of units (for METHOD = firs, last or zigz)

Description

This directive was renamed HREDUCE in Release 14, but the original name of REDUCE is currently still retained as a synonym. However, it may be removed in a future release.

Sometimes you may want to regard an n-by-n similarity matrix S as being partitioned into b-by-b rectangular blocks. You might then want to form a reduced matrix of similarities, between the different blocks instead of between the individual units. To do this you have to arrange for each of the b^2 blocks of the full matrix to be replaced by a single value. Each diagonal block must be replaced by unity. The METHOD option specifies how to replace the off-diagonal blocks, for example the maximum, minimum or mean similarity within the block. The *zigzag* method (Rayner 1966) is relevant in particular when the data consist of b soil samples for each of which information is recorded on several soil horizons, possibly different in the different samples. The method recognizes that certain horizons might be absent from some soil samples; this leads to finding successive optimal matches, conditional on the constraint that one horizon cannot match a horizon that has already been assigned to a higher level; after finding these optima, an average is taken for each horizon.

The SIMILARITY parameter specifies the similarity matrix for the full set of n observations; this must be present and have values. The REDUCEDSIMILARITY parameter specifies an identifier for the reduced similarity matrix, of order b; this will be declared implicitly if you have not declared it already. The factor that defines the classification of the units into groups must be specified by the GROUPS parameter. The units can be in any order, so that for example the units of the first group need not be all together nor given first. The labels of the factor label the reduced similarity matrix.

The PERMUTATION parameter, if present, must specify a variate. It defines the ordering of samples within each group, and so must be specified for methods first, last and zigzag. Within each group, the unit with the lowest value of the permutation variate is taken to be the first sample, and so on. Genstat will, if necessary, use a default permutation of one up to the number of rows of the similarity matrix.

If you set option PRINT=similarities, the values of the reduced symmetric matrix are printed, as percentages.

Options: PRINT, METHOD.
Parameters: SIMILARITY, REDUCEDSIMILARITY, GROUPS, PERMUTATION.

Reference

Rayner, J.H. (1966). Classification of soils by numerical methods. *Journal of Soil Science*, **17**, 79-92.

See also

Directive: HREDUCE.

Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

REFORMULATE

Modifies a formula or an expression to operate on a different set of data structures.

Options

OLDFORMULA = *formula* or *expression structures*
: Original formula or expression

NEWFORMULA = *formula* or *expression structures*
: New formula or expression, modified to operate on the new structures

Parameters

OLDSTRUCTURE = *identifiers*
: Data structures in the OLDFORMULA to be replaced in the NEWFORMULA

NEWSTRUCTURE = *identifiers*
: Identifier of the new data structure to replace each OLDSTRUCTURE

Description

The REFORMULATE directive allows you to modify a formula or expression to operate on a different set of data structures. The original formula or expression is specified by the OLDFORMULA option, and the new formula or expression is specified by the NEWFORMULA option. If NEWFORMULA is not specified, the new formula or expression replaces the old one in OLDFORMULA. The data structures to be replaced in OLDFORMULA are listed by the OLDSTRUCTURE parameter, and the corresponding data structures for NEWFORMULA are provided by the NEWSTRUCTURE parameter.

The example below shows how you could convert formula A*B (stored in Old) into formula Y*Z (stored in New).

```
FORMULA     [VALUE=A*B] Old
REFORMULATE [OLDFORMULA=Old; NEWFORMULA=New]\
            OLDSTRUCTURE=A,B; NEWSTRUCTURE=Y,Z
```

Options: OLDFORMULA, NEWFORMULA.
Parameters: OLDSTRUCTURE, NEWSTRUCTURE.

See also

Directives: EXPRESSION, FORMULA, FARGUMENTS, FCLASSIFICATION, SET2FORMULA.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

RELATE

Relates the observed values on a set of variates or factors to the results of a principal coordinates analysis (synonym of PCORELATE).

Options

COORDINATES = *matrix*	Points in reduced space; no default i.e. this option must be specified
NROOTS = *scalar*	Number of latent roots for printed output; default * requests them all to be printed

Parameters

DATA = *variates* or *factors*	The data variables
TEST = *string tokens*	Test type, defining how each variable is treated in the calculation of the similarity between each unit (simplematching, jaccard, russellrao, dice, antidice, sneathsokal, rogerstanimoto, cityblock, manhattan, ecological, euclidean, pythagorean, minkowski, divergence, canberra, braycurtis, soergel); default * ignores that variable
RANGE = *scalars*	Range of possible values of each variable; if omitted, the observed range is takenz

Description

This directive was renamed PCORELATE in Release 14, but the original name of RELATE is currently still retained as a synonym. However, it may be removed in a future release.

One way of interpreting the principal coordinates obtained from a similarity matrix by PCO is by relating them to the original data variables. For each coordinate and each data variable, an F-statistic can be computed as if the variable and the coordinate vector were independent. This is not the case but, although the exact distribution of these pseudo F-values is not known, they do serve to rank the variables in order of importance of their contribution to the coordinate vector.

The DATA parameter lists the variables (variates or factors) that are to be related to the PCO results and the TEST parameter indicates their "type" as in the FSIMILARITY directive. The RANGE parameter contains a list of scalars, one for each variable in the DATA list, allowing you to standardize quantitative variates.

Qualitative variables (variates or factors with TEST settings simplematching - rogerstanimoto) are treated as grouping factors, and the mean coordinate for each group is calculated. Only 10 groups are catered for; group levels above 10 are combined. The pseudo F-statistic gives the between-group to within-group variance ratio. Missing values are excluded.

Quantitative variables (i.e. variates with other settings) are grouped on a scale of 0-10 (where zero signifies a value up to 0.05 of the range), and mean coordinates for each group are calculated. The printed pseudo F statistic is for a linear regression of the principal coordinate on the ungrouped data variate, after standardizing the data variate to have unit range; the regression coefficient is also printed.

The COORDINATES option must be present and must be a matrix. This represents the units in reduced space. Usually the coordinates will be from a principal coordinates analysis. The number of rows of the matrix must match the number of units present in the variables, taking account of any restriction.

The output from RELATE can be extensive. You may not be interested in relating the variables to the higher dimensions of the principal coordinates analysis even though you may have saved these in the coordinate matrix. The NROOTS option can request that results for only some of the

dimensions are printed. If NROOTS is not specified, RELATE prints information for all the saved dimensions: that is, for the number of columns of the coordinates matrix.

Options: COORDINATES, NROOTS.
Parameters: DATA, TEST, RANGE.

See also
Directive: PCORELATE.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

REML

Fits a variance-components model by residual (or restricted) maximum likelihood.

Options

PRINT = *string tokens*	What output to present (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default mode, comp, Wald, cova
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff
WEIGHTS = *variate*	Weights for the analysis; default * implies all weights 1
MVINCLUDE = *string tokens*	Whether to include units with missing values in the explanatory factors and variates and/or the y-variates (explanatory, yvariate); default * i.e. omit units with missing values in either explanatory factors or variates or y-variates
SUBMODEL = *formula*	Defines a submodel of the fixed model to be assessed against the full model (for METHOD=Fisher only)
RECYCLE = *string token*	Whether to reuse the results from the estimation when printing or assessing a submodel (yes, no); default no
RMETHOD = *string token*	Which random terms to use when calculating RESIDUALS (final, all, notspline); default fina
METHOD = *string token*	Indicates whether to use the standard Fisher-scoring algorithm or the new AI algorithm with sparse matrix methods (Fisher, AI); default AI
MAXCYCLE = *scalar*	Limit on the number of iterations; default 30
TOLERANCES = *variate*	Tolerances for matrix inversion; default * i.e. appropriate default values
PARAMETERIZATION = *string token*	Parameterization to use for the variance component estimation (gammas, sigmas); default * i.e. use whichever is most appropriate for model
CFORMAT = *string token*	Whether printed output for covariance models gives the variance matrices or the parameters (variancematrices, parameters); default vari
FMETHOD = *string token*	Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the algorithm

Parameters

Y = *variates*	Variates to be analysed
RESIDUALS = *variates*	Residuals from each analysis
FITTEDVALUES = *variates*	Fitted values from each analysis
EXIT = *scalar*	Exit status of the fit (0 if successful)
SAVE = *REML save structures*	Saves the details of each analysis for use in subsequent

VDISPLAY and VKEEP directives

Description

REML estimates the treatment effects and variance components in a linear mixed model: that is, a linear model with both fixed and random effects. The model to be fitted is specified using the VCOMPONENTS directive, covariance models for the random effects can be defined using the VSTRUCTURE directive, and the VRESIDUAL directive can define covariance models for the residual term or specify the residual term for the individual experiments in a meta-analysis. Further output can be produced following REML using VDISPLAY, and output can be saved in Genstat data structures using VKEEP. REML can be used in situations where you would normally use ANOVA but have unbalanced or correlated data, or where you would normally use linear regression, but have more than one source of variation or correlation in the data.

REML can be used to analyse data from a wide variety of applications. It can obtain information on sources and sizes of variability in data sets. This can be of interest where the relative size of different sources of variability must be assessed, for example to identify the least reliable stages in an industrial process, or to design more effective experiments. REML also provides efficient estimates of treatment effects in unbalanced designs with more than one source of error. It can be used to provide estimates of treatment effects that combine information from all the strata of a partially balanced design, or to combine information over similar experiments conducted at different times or in different places. You can thus obtain estimates that make use of the information from all the experiments, as well as the separate estimates from each individual experiment. Examples from several different areas of application can be found in Robinson (1987). The facilities for estimation of covariance models allow estimates of treatment effects and standard errors to be obtained using an appropriate variance model and taking account of the correlation structure of the data.

The method of residual maximum likelihood (REML) was introduced by Patterson & Thompson (1971). It was developed in order to avoid the biased variance component estimates that are produced by ordinary maximum likelihood estimation: because maximum likelihood estimates of variance components take no account of the degrees of freedom used in estimating treatment effects, they have a downwards bias which increases with the number of fixed effects in the model. This in turn leads to under-estimates of standard errors for fixed effects, which may lead to incorrect inferences being drawn from the data. Estimates of variance parameters which take account of the degrees of freedom used in estimating fixed effects, like those generated by ANOVA in balanced data sets, are more desirable.

Once a mixed model has been specified (using VCOMPONENTS) and any covariance structures have been defined (using VSTRUCTURE and VRESIDUAL) you can fit the model to the data (the y-variates) using the REML directive.

The Y parameter lists the variates that are to be modelled. For example, given appropriate factor definitions, the following command sets up a model and analyses the data held in variate Yield:

```
VCOMPONENTS [FIXED=Nitrogen*Variety] RANDOM=Block/Wplot/Splot
REML Yield; FITTED=Fit; RESIDUALS=Res
```

The FITTEDVALUES and RESIDUALS parameters allow you to store the fitted values and residuals from the fitted model – above they are stored in variates Fit and Res. The EXIT parameter saves the "exit status" of each analysis. This is set to zero if it was completed successfully; for details of the other codes, see VKEEP. The SAVE parameter can be used to name the REML save structure for use with later VKEEP and VDISPLAY directives.

The three options PRINT, PTERMS and PSE all control the printed output. The PRINT option selects the output to be displayed:

model	description of model fitted

`components`	estimates of variance components and estimated parameters of covariance models
`effects`	estimates of parameters α and β, the fixed and random effects
`means`	predicted means for factor combinations
`stratumvariances`	approximate stratum variances from a decomposition of the information matrix for the variance components (available only when `METHOD=Fisher`)
`monitoring`	monitoring information at each iteration
`vcovariance`	variance-covariance matrix of the estimated components
`deviance`	deviance of the fitted model ($-2 \times$ log-likelihood RL) plus deviance of submodel when fitted
`waldtests`	Wald tests for all fixed terms in model
`missingvalue`	estimates of missing values
`covariancemodels`	estimated covariance models

The default setting of `PRINT=model,components,Wald,cova`, gives a description of the model and covariance models that have been fitted, plus estimates of the variance components and the Wald tests. By default if tables of means and effects are requested, tables for all terms in the fixed model are given together with a summary of the standard error of differences between effects/means. Options `PTERMS` and `PSE` can be used to change the terms or obtain different types of standard error. For example,

```
VCOMPONENTS [FIXED=Nitrogen*Variety] RANDOM=Block/Wplot/Splot
REML [PRINT=means; PTERMS=Nitrogen.Variety; \
     PSE=allestimates] Yield
```

means that a `Nitrogen` by `Variety` table of predicted means will be produced with a standard error for each cell.

The `FMETHOD` option controls whether to accompany the Wald tests for fixed effects with approximate F statistics and corresponding numbers of residual degrees of freedom. The computations, using the method devised by Kenward & Roger (1997), can be time consuming with large or complicated models. So, with the default setting `FMETHOD=automatic`, Genstat assesses the model itself and decides automatically whether to do the computations and which method to use. The other settings allow you to control what to do yourself:

`none`	no F statistics are produced;
`algebraic`	F statistics are calculated using algebraic derivatives (which may involve large matrix calculations);
`numerical`	F statistics are calculated using numerical derivatives (which require an extra evaluation of the mixed model equations for every variance parameter).

The `CFORMAT` option controls the type of output produced for the estimated covariance models. The default setting, `variancematrices`, produces the variance-covariance matrices for the components, whereas the setting `parameters` prints their parameters.

The `MVINCLUDE` option allows the inclusion of units with missing values. By default, units where there is a missing value in the y-variate or in any of the factors or variates in the model terms are excluded. The setting `explanatory` allows units with missing values in factors or variates in the model to be included. For missing covariate values, this is equivalent to substituting the mean value. The setting `yvariate` includes units with missing values in the y-variate. This can be useful to retain the balanced structure of the data for use with direct product covariance matrices (see `VSTRUCTURE`), or to produce predictions of data values for given values of explanatory factors and/or variates.

The `WEIGHTS` option can be used to specify a weight for each unit in the analysis. This is useful when it is suspected that the size of the random error varies between units. For example,

if the random error for unit i is known to have variance $v_i\sigma^2$, a weight variate should be used containing values $w_i=1/v_i$.

The RMETHOD option controls the way in which residuals and fitted values are formed. For the default setting RMETHOD=final, the fitted values are calculated from all the fixed and random effects. The residuals are the difference between the data and the fitted values and, in this case, are estimates of the *units* random error and can be used to check the Normality and variance homogeneity assumptions for the random error. To get fitted values constructed from the fixed terms alone, omitting all random terms, the setting RMETHOD=all must be used. The setting RMETHOD=notspline means that the residuals will be formed from all the random effects, excluding spline terms.

Option SUBMODEL is used to specify a submodel of the fixed model (but only applies when METHOD=Fisher). This model will be fitted as well as the full fixed model, using a slightly modified version of the algorithm, and the difference in deviances between the full and submodel can be used as a likelihood-based test to assess the importance of the fixed terms dropped from the full model, as described by Welham & Thompson (1997). Once the full model has been fitted, the RECYCLE option can be used to test a series of submodels of the fixed model. If option RECYCLE=yes is set, then only the estimation for the submodel is performed. Information for the full fixed model is picked up from the corresponding save structure. When the RECYCLE option is set, only the deviance and model settings of PRINT can be used.

The METHOD option specifies whether to use the AI (Average Information) algorithm (Gilmour *et al*. 1995) with sparse matrix methods to maximize the residual likelihood, or Fisher scoring with full matrix manipulation. By default, the sparse Average Information algorithm is used. The AI algorithm generally runs faster per iteration than Fisher scoring and uses much less workspace, but it may require slightly more iterations to reach convergence. When sparse matrix methods are used, standard errors of differences will not be available for random effects, although standard errors are available. Note that when METHOD=AI, the SUBMODEL and RECYCLE options do not apply. The WORKSPACE option (default 1) specifies the number of blocks of internal memory to be allocated for use by the estimation algorithm when METHOD=AI.

Option MAXCYCLE can be used to change the maximum number of iterations performed by the algorithm from the default of 30.

The TOLERANCES option gives tolerances for three matrix inversions. The first two values are matrix inversion tolerances for the information matrix and the mixed model equations respectively and take the value 10^{-5} by default. The third value is used to detect zero frequency counts for factor combinations in the mixed model equations: 10^{-6} is used by default.

Options: PRINT, PTERMS, PSE, WEIGHTS, MVINCLUDE, SUBMODEL, RECYCLE, RMETHOD, METHOD, MAXCYCLE, TOLERANCES, PARAMETERIZATION, CFORMAT, WORKSPACE.
Parameters: Y, RESIDUALS, FITTEDVALUES, EXIT, SAVE.

Action with RESTRICT

Any of the y-variates or any of the factors or variates in the fixed and random models (defined by VCOMPONENTS) may be restricted to indicate that only a subset of the units is to be used in the analysis. However, if more than one of these vectors is restricted, all must be restricted to the same set of units. Any restrictions on the variates supplied to save residuals or fitted values are ignored.

References

Gilmour, A.R., Thompson, R. & Cullis, B. (1995). AIREML, an efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, **51**, 1440-1450.

Kenward, M.G. & Roger, J.H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics*, **53**, 983-997.

Patterson, H.D. & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, **58**, 545-554.

Robinson, D.L. (1987). Estimation and use of variance components. *The Statistician*, **36**, 3-14.

Welham, S.J. & Thompson, R. (1997). Likelihood ratio tests for fixed model terms using residual maximum likelihood. *Journal of the Royal Statistical Society, Series B*, **59**, 701-714.

See also

Directives: VCOMPONENTS, VSTRUCTURE, VRESIDUAL, VDISPLAY, VPREDICT, VKEEP, VPEDIGREE, VCYCLE, VSTATUS.

Procedures: VAIC, VALLSUBSETS, VBOOTSTRAP, VCHECK, VCRITICAL, VFRESIDUALS, VGRAPH, VPLOT, VDEFFECTS, VDFIELDRESIDUALS, VFIXEDTESTS, VFLC, VFPEDIGREE, VFUNCTION, VHERITABILITY, VLSD, VMETA, VMCOMPARISON, VPERMTEST, VPOWER, VRACCUMULATE, VRCHECK, VRFIT, VRMETAMODEL, VRPERMTEST, VSAMPLESIZE, VSCREEN, VSOM, VSPREADSHEET, VSURFACE, VTCOMPARISONS, VABLOCKDESIGN, VAMETA, VAROWCOLUMNDESIGN, VASERIES, VALINEBYTESTER, VLINEBYTESTER, FCONTRASTS, FDIALLEL, AOVANYHOW.

Genstat Reference Manual 1 Summary sections on: REML analysis of linear mixed models, Analysis of variance.

RENAME

Assigns new identifiers to data structures.

No options

Parameters

OLDIDENTIFIER = *identifiers*	Specifies the data structures to rename
NEWIDENTIFIER =*identifiers*	Specifies a new identifier for each data structure

Description

RENAME allows you to assign a different identifier to a data structure. For example, if you put

```
RENAME OLDIDENTIFIER=A; NEWIDENTIFIER=B
```

the data structure previously known as A would be renamed to have the identifier B, and the data structure previously known as B would lose its identifier and become unnamed. The identifier A would then no longer belong to anyone (and could if required be reused).

In the simplest situations, the first appearance of the new identifier will be in the RENAME command. So there will be no consequences from the fact that the "orphan" data structure that it previously identified becomes unnamed.

If the identifier has already been used, the orphan data structure will be deleted, unless it is found to belong to another (named) data structure. So, for example, if the full program was

```
SCALAR B; VALUE=1
POINTER [VALUES=B] Q
RENAME OLDIDENTIFIER=A; NEWIDENTIFIER=B
```

the scalar 1 would survive as the first element of the pointer Q. So it could still be referred to as Q[1], although of course no longer as B. You would get the same effect be specifying

```
RENAME OLDIDENTIFIER=A; NEWIDENTIFIER=Q[1]
```

as RENAME looks only for the (named) identifier of the data structure specified by NEWIDENTIFIER. So, in this case, A takes over the identifier B of Q[1]. If Q[1] did not have a separate identifier of its own, A would become unnamed. (So this provides a way of removing the identifier of a pointer element.)

You can also specify a pointer element for the setting of OLDIDENTIFIER and, again, RENAME will operate only on its identifier (if it has one). For example, in the program

```
SCALAR C; VALUE=7
POINTER [NVALUES=2] P
RENAME OLDIDENTIFIER=P[1]; NEWIDENTIFIER=C
```

the pointer element P[1] gains the identifier C, and so can be referred to as C in future (as well as P[1]).

So, to summarize, for the data structures specified by both the OLDIDENTIFIER and NEWIDENTIFIER parameters, RENAME ignores any memberships that they may have of pointers, or e.g. as classifying factors of a table, or as levels or labels vectors of factors. It operates only on their own identifiers, reassigning the one (if any) belonging to the NEWIDENTIFIER data structure to become the identifier of the data structure specified by the OLDIDENTIFIER parameter.

Finally note that, if either OLDIDENTIFIER or NEWIDENTIFIER is set to a dummy, RENAME will operate on the data structure to which it points, not on the dummy itself (i.e. dummies are always substituted). So, this allows you to rename data structures in your main program from inside a procedure.

Options: none.
Parameters: OLDIDENTIFIER, NEWIDENTIFIER.

See also

Directives: DUMMY, POINTER, PROCEDURE.

Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Data structures.

RESTRICT

Defines a restricted set of units of vectors for subsequent statements.

No options

Parameters

VECTOR = *vectors*	Vectors to be restricted
CONDITION = *expression*	Logical expression defining the restriction for each vector; a zero (false) value indicates that the unit concerned is not in the set
SAVESET = *variates*	List of the units in each restricted set
NULL = *scalars*	Indicator for each restricted set, set to 1 or 0 according to whether or not it contains no units

Description

The RESTRICT directive defines a *restriction* on the units of a vector, so that future operations will involve only a subset of the units.

The VECTOR parameter specifies the vector or vectors that are to be restricted. These can be variates, factors or texts, but all the vectors listed must be of the same length.

The CONDITION parameter specifies a logical expression which indicates which units of the vectors are in the defined subset. For example,

```
VARIATE [VALUES=1,2,3,2,3,4,3,4,5] V
RESTRICT V; CONDITION=V.EQ.2
```

restricts the vector V to those units with the value 2. Genstat evaluates the expression to generate internally a variate of zeroes and ones, of the same length as the vectors being restricted. A zero value indicates that the corresponding unit is to be excluded. The logical expression can involve any vector of the same length as the vector to be restricted. For example, to restrict variate V and text T to the units with levels 1 or 2 or 4 of factor F, you could use the statement

```
RESTRICT V,T; CONDITION=(F.LE.2).OR.(F.EQ.4)
```

When using a text to define a restriction, remember that you cannot use logical operators like .EQ. and .NE. Instead you should use operators .IN., .NI., .EQS. and .NES.:

```
TEXT [VALUES=London,Madrid,Nairobi,Ottawa,Paris,Quito,Rome]\
     City
&    [VALUES=London,Madrid,Paris,Rome] Europe
RESTRICT City; CONDITION=City.IN.Europe
```

restricts the text City to lines 1, 2, 5 and 7 only.

Of course, the expression may just contain a single variate of the of the same length as the vectors to be restricted. Again a zero indicates that the corresponding unit in the vector to be restricted is excluded, while any non-zero entry causes inclusion. Thus the restriction above on the text City could also be specified by

```
RESTRICT T; CONDITION=!(1,1,0,0,1,0,1)
```

The same effect can be achieved by using the EXPAND function:

```
RESTRICT City; CONDITION=EXPAND(!(1,2,5,7))
```

Another function that may be useful is RESTRICTION; this allows you to generate a variate of ones and zeros indicating the units to which a vector is currently restricted. It thus provides a very convenient way of transferring a restriction from one vector to another. For example,

```
RESTRICT Timezone,Distance; CONDITION=RESTRICTION(City)
```

restricts the vectors Timezone and Distance to the same units as those to which City is currently restricted.

Finally, if you omit the CONDITION parameter, this removes any restrictions on the vectors are removed. For example

```
RESTRICT City,Timezone,Distance
```

removes any restrictions that have been set on City, Timezone and Distance.

Note that if the vectors used in the CONDITION expression are themselves restricted these restrictions will remain in force during the current calculation of the condition. A danger here, therefore, is that you may accidentally end up restricting out all the elements of a vector by using RESTRICT repeatedly. The safest way to avoid this is to remove the restrictions on any vectors to be used in the CONDITION expression before you use them to restrict vectors in some different way.

The SAVESET parameter can be used to save the numbers of the units that are in the restricted set. These are saved in a variate with one value for each unit retained by the restriction. Thus, if the example above with variate V were to become

```
VARIATE [VALUES=1,2,3,2,3,4,3,4,5] V
RESTRICT V; CONDITION=V.EQ.2; SAVESET=S
```

S would be created as a variate of length 2, with values 2 and 4.

The NULL parameter can specify a list of scalars, one for each vector in the VECTOR list, that will be set to one if its restricted set contains no units; otherwise it is set to zero. Also, when NULL set, RESTRICT suppresses the warnings that it normally gives if a restricted set is null.

Not all directives take account of RESTRICT. For those that do, usually only one vector in the list of parameters has to be restricted for the directive to treat them all as being restricted in the same way. A fault is reported if any vectors in such a list are restricted in different ways.

Options: none.
Parameters: VECTOR, CONDITION, SAVESET.

See also
Directives: EXPRESSION, CALCULATE.
Procedures: FRESTRICTEDSET, SUBSET.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

RESUME

Restarts a recorded job.

Options

CHANNEL = *scalar*	Channel number of the backing-store file where the information was dumped; default 1
CLOSE = *string token*	Whether to close the file afterwards (yes, no); default no

No parameters

Description

RESUME recovers the information stored by a previous RECORD statement so that you can continue your use of Genstat as though nothing had happened in between. Genstat deletes all the data structures that were created in the current job prior to RESUME, and reinstates the data structures that were available in Genstat at the time the RECORD statement took place. In addition, the current graphics settings are replaced by those that were in force when RECORD was used, but any external files that are attached to Genstat remain unaffected.

If the RECORD directive was used within a procedure or a FOR loop, the job is not resumed at that point. Instead, it restarts at the statement after the procedure call, or after the outermost ENDFOR statement.

The CHANNEL option specifies the channel to which the file has been connected (this can be done using the OPEN directive). You can set the CLOSE option to yes to close the file after the information has been recovered.

Options: CHANNEL, CLOSE.
Parameters: none.

See also

Directives: RECORD, OPEN.
Genstat Reference Manual 1 Summary section on: Input and output.

RETRIEVE

Retrieves structures from a subfile.

Options

CHANNEL = *scalar*	Specifies the channel number of the backing-store or procedure-library file containing the subfile (FILETYPE settings 'back' or 'proc'); default 0 (i.e. the workfile) for FILETYPE=back, no default for FILETYPE=proc, not relevant with other FILETYPE settings
SUBFILE = *identifier*	Identifier of the subfile; default SUBFILE
LIST = *string token*	How to interpret the list of structures (inclusive, exclusive, all); default incl
MERGE = *string token*	Whether to merge structures with those already in the job (yes, no); default no, i.e. a structure whose identifier is already in the job overwrites the existing one, unless it has a different type
FILETYPE = *string token*	Indicates the type of file from which the information is to be retrieved (backingstore, procedurelibrary, siteprocedurelibrary, Genstatprocedurelibrary); default back

Parameters

IDENTIFIER = *identifiers*	Identifiers to be used for the structures after they have been retrieved
STOREDIDENTIFIER = *identifiers*	Identifier under which each structure was stored

Description

You can recover information from a subfile of a backing-store file using the RETRIEVE directive. The CHANNEL option specifies the backing-store file, and the SUBFILE option indicates the subfile. Both these options can be omitted; by default the file will be the workfile, and the subfile will be called SUBFILE.

When you retrieve a structure Genstat may also retrieve a chain of associated structures: that is, all the structures to which it points, and the structures to which they point, and so on. For example, suppose you store the three structures with identifiers T, V and F, along with an unnamed structure storing information about T, in a subfile called SUBFILE in backing-store file FILE1:

```
OPEN 'FILE1'; CHANNEL=1; FILETYPE=backingstore
TEXT [VALUES=a,b,c] T
VARIATE V; EXTRA=T
FACTOR [LABELS=T] F
STORE [CHANNEL=1] T,V,F
```

Then the statement

```
RETRIEVE [CHANNEL=1] V
```

will retrieve not only V but also T (which was associated with T by the EXTRA parameter of the VARIATE statement), and the unnamed structure that is associated with T. The structures V, T and the unnamed structure, are said to be a *complete set* from the subfile.

The IDENTIFIER parameter specifies the structures to be retrieved. You can use the STOREDIDENTIFIER parameter to give a structure a different name from the one within the subfile. For example

```
RETRIEVE IDENTIFIER=Weeks; STOREDIDENTIFIER=Time
```

You are not allowed to give identical identifiers to two retrieved structures, nor are you allowed

to have the same identifier referring to a structure of one type in a subfile, and to a structure of a different type in your job.

As with STORE, if you want to rename only some of the structures, you can either respecify the existing identifier, or insert * at the appropriate point in the STOREDIDENTIFIER list.

Genstat knows whether you are retrieving a procedure by the type of file that you are accessing, as set by the FILETYPE option. You are not allowed to rename a procedure as a suffixed identifier or as the name of a directive.

You can even rename a structure so that it is unnamed in the job. Suppose, for example, that a structure T already exists within Genstat, and that you want to retrieve the variate V stored in the file FILE1 above. Then, as we have seen, the structure T will also be retrieved. However, you can avoid the existing structure T in the job being overwritten by making the retrieved version of T unnamed:

```
OPEN 'FILE1'; CHANNEL=1; FILETYPE=backingstore
RETRIEVE [CHANNEL=1] V,!T(a); STOREDIDENTIFIER=V,T
```

The value, a, of the unnamed text !T(a) will be replaced by the values stored for T, and this unnamed text will become the EXTRA text for V. Alternatively you could rename T to be Tnew by

```
RETRIEVE [CHANNEL=1] V,TNew; STOREDIDENTIFIER=V,T
```

When you are retrieving a suffixed identifier, Genstat matches the numerical suffix only, and not the whole structure that is denoted by the identifier. For example, suppose pointer P stored in a subfile points to structures with identifiers A, B, C and D, and that P has numerical suffixes 1 to 4 respectively. Also suppose that in your current job, you have never mentioned pointer P either directly or indirectly. Then the statement

```
RETRIEVE [CHANNEL=1] P[2]
```

will retrieve the structure B from backing store but, as it has not been referenced only as P[2] in the RETRIEVE statement, the identifier B will not be recovered and it will be known only as P[2] within Genstat.

A structure that you are retrieving from a subfile may sometimes overwrite the values of an existing structure in your program. If this structure is a pointer or a compound structure, the existing suffixes will be overwritten by those of the stored structure, so some existing structures with suffixed identifiers may in effect be lost. For example, suppose that userfile FILE2 contains a pointer P, with suffixes 1 and 2 pointing to structures A and B. If we set up a variate P[3], and then retrieve the pointer P

```
OPEN 'FILE2'; CHANNEL=1; FILETYPE=backingstore
VARIATE [VALUES=1...6] P[5,6,7]
RETRIEVE [CHANNEL=1] P
```

P will now have suffixes 1 and 2 pointing to A and B, but the variate P[3] will have been lost.

The LIST option controls how the IDENTIFIER list is interpreted. The default setting inclusive simply retrieves the structures that have been listed. Alternatively, if you set LIST=all Genstat will retrieve all the structures in the subfile that have identifiers and whose types have been defined. Finally, you can see LIST=exclusive to retrieve everything in the subfile that you have not listed in the IDENTIFIER parameter. Note, though, that some of the structures in the IDENTIFIER list may be retrieved if they are needed to complete the set of structures to be retrieved. If you use this setting, the STOREDIDENTIFIER parameter is ignored.

The FILETYPE option specifies whether you wish to retrieve information from backing store files that have been attached as normal backing store files or as procedure libraries by the OPEN directive, or from Genstat Procedure library or from the site procedure library. The CHANNEL setting is ignored if the siteprocedurelibrary or Genstatprocedurelibrary settings are used. The source code of the procedures in the Genstat Procedure library can be accessed using the LIBEXAMPLE procedure.

Normally when you retrieve a complete subset of structures, Genstat overwrites all structures in the job that have the same identifier (after any renaming). As a result, some other structures already in the job may become inconsistent and will be destroyed. You can avoid this happening by setting the MERGE option to yes. Genstat then does not overwrite any structures with the same name and type. However, a consequence is that some of the retrieved structures may now be inconsistent and thus need to be destroyed in the program (although they will of course remain in the subfile).

Options: CHANNEL, SUBFILE, LIST, MERGE, FILETYPE.
Parameters: IDENTIFIER, STOREDIDENTIFIER.

See also
Directives: STORE, CATALOGUE, MERGE, OPEN, RECORD, RESUME.
Genstat Reference Manual 1 Summary section on: Input and output.

RETURN

Returns to a previous input stream (text vector or input channel).

Options

`NTIMES` = *scalar*	Number of streams to ascend; default 1
`CLOSE` = *string token*	Whether to close the channel (or text) after the return (`yes`, `no`); default `no`
`DELETE` = *string token*	Whether to delete the text or the file to which the channel was attached (only relevant if `CLOSE=yes`) after the return (`yes`, `no`); default `no`

Parameter

expression	Logical expression controlling whether or not to return to the previous input stream; default 1 (i.e. true)

Description

In its simplest form, you can type

```
RETURN
```

to make Genstat stop taking statements from the current input channel and to go back to the channel that was previously active, and contained the `INPUT` statement that switched to the current file. Input then continues from the line following the original `INPUT` statement, but a marker is left in the channel that contains the `RETURN` statement, so that you can use `INPUT` to continue from the next line after `RETURN` later in your programme.

Sometimes you may want to return only if a particular condition is satisfied, for example if you have discovered that the data are unsatisfactory for whatever operations occur later in the file. To do this, you set the parameter to an appropriate logical expression; this must return a scalar result, which is interpreted as *true* if it is equal to 1, and *false* otherwise. For example

```
RETURN MIN(Height)<0
```

If you have use `INPUT` several times, you may wish to return through several channels. The `NTIMES` option can be set to a number, or a scalar, to control how many returns take place. For example, with input starting on channel 1, supposing you had used `INPUT 2` to switch to a file on channel 2, and then `INPUT 3` to switch to a further file (on channel 3). If this file then contained the statement `RETURN [NTIMES=2]` you would return to channel 1. You can never return from input channel 1, so if you set `NTIMES` to a number greater than the number of currently active input channels, Genstat simply returns to channel 1.

You can set option `CLOSE=yes` to close the file; also, if you do have `CLOSE=yes`, you can set `DELETE=yes` to delete the file.

If Genstat meets the end of the file on the current input channel, it will try to return control to the channel from which it was called. This is called an *implicit return*. The channel is closed automatically when this happens, and a warning message will be printed.

In order to maintain control over the different input channels, and know where to go after a `RETURN`, Genstat keeps an internal stack of input channels. Suppose you specify channel *k*, by typing `INPUT` *k*. There are three possible actions:

(a) if *k* is the current input channel, the statement is ignored;
(b) if *k* is not in the stack, it is added to it;
(c) if *k* is already in the stack (that is, the current state is: $1 \rightarrow ... \rightarrow k \rightarrow k_1 \rightarrow k_2 \rightarrow ... \rightarrow k_n$) then the intermediate channels k_1 ... k_n are suspended at their current positions and removed from the stack.

Input then switches to channel *k*, taking statements from the beginning of the file if it has never been used before, or from the point at which it was last suspended. Subsequent `INPUT` statements will re-start the other channels from where they were suspended. When a `RETURN` statement is

used, Genstat steps back NTIMES through the stack, removing any intermediate channels from the stack. This means that, using the above representation of the input stack, if channel k_n contained the statement INPUT k_2 and channel k_2 then had a RETURN, this would return to channel k_1.

If you use ## to execute macros, these are treated in the same way as input channels and added to the input stack. You can use INPUT to temporarily halt a macro and switch to a file, and RETURN to get back to the macro.

Options: NTIMES, CLOSE, DELETE
Parameter: unnamed.

See also
Directives: INPUT, CALCULATE.
Genstat Reference Manual 1 Summary section on: Input and output.

RFUNCTION

Estimates functions of parameters of a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (estimates, se, correlations); default esti,se
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
CALCULATION = *expression structures*	Calculation of functions involving nonlinear and/or linear parameters; no default
SE = *variate*	To save approximate standard errors; default *
VCOVARIANCE = *symmetric matrix*	To save approximate variance-covariance matrix; default *
SAVE = *identifier*	Specifies save structure of regression model; default * i.e. that from last model fitted

Parameter

scalars	Identifiers of scalars assigned values of the functions by the calculations

Description

The RFUNCTION directive provides estimates of functions of parameters in regression models, together with approximate standard errors and correlations. It can be used after any regression model except after fitting standard curves with separate nonlinear parameters for each level of a factor (option NONLINEAR=separate in FITCURVE, ADD, DROP and SWITCH). However, if there are any linear parameters in a general nonlinear model for which standard errors have not been estimated, standard errors and correlations cannot be estimated for functions that depend on those parameters.

The functions are defined by the expressions supplied by the CALCULATION option of RFUNCTION. These define how to calculate the function from the values of the parameters.

In linear and generalized linear models, the parameters have no identifiers associated with them. You should then refer to each parameter by using a text structure containing the name of the parameter as displayed, for example, by the option PRINT=estimates of the FIT directive. The text structure can, of course, just be a string, for example 'Constant'. However, it must match exactly, including case, the name displayed by FIT.

Unless initial values have been specified using the RCYCLE directive, parameters in standard curves (fitted by FITCURVE) usually also have no identifiers and so should be referred to using texts as for linear regression models. However, in nonlinear models (fitted by FITNONLINEAR) identifiers are specified for the nonlinear parameters using RCYCLE. Names can be specified for the linear parameters of nonlinear models using the LINEARPARAMETERS option of RCYCLE; if not, texts must be used as in linear regression models.

The parameter of RFUNCTION provides a list of scalars that are to hold the estimated values of the functions. These need not be declared in advance, but will be defined automatically if necessary. The CALCULATION option specifies a list of one or more expressions to define the calculations necessary to evaluate the functions from the parameters of the model, and place the results into the scalars.

The PRINT option controls output as usual. By default, the estimates of the function values are formed – as could be done simply by a CALCULATE statement using the expressions if the parameters were available in scalars. In addition, approximate standard errors are calculated,

using a first-order approximation based on difference estimates of the derivatives of each function with respect to each parameter. Approximate correlations can also be requested.

The SE and VCOVARIANCE options allow standard errors and the approximate variance-covariance matrix of the functions to be stored; the estimates of the functions themselves are automatically available in the scalars listed by the parameter of RFUNCTION. The SAVE option specifies which fitted model is to be used, as in the RDISPLAY and RKEEP directives.

Options: PRINT, CHANNEL, CALCULATION, SE, VCOVARIANCE, SAVE.
Parameter: unnamed.

See also
Directives: EXPRESSION, FIT, FITCURVE, FITNONLINEAR.
Procedures: NLAR1, FIELLER.
Genstat Reference Manual 1 Summary section on: Regression analysis.

RKEEP

Stores results from a linear, generalized linear, generalized additive or nonlinear model.

Options

EXPAND = *string token*	Whether to put estimates in the order defined by the maximal model for linear or generalized linear models (yes, no); default no
DISPERSION = *scalar*	Dispersion parameter to be used as estimate for variability in s.e.s; default as set in the MODEL directive
RMETHOD = *string token*	Type of residuals to form if parameter RESIDUALS is set (deviance, Pearson, simple); default as set in MODEL
DMETHOD = *string token*	Basis of estimate of dispersion, if not fixed by DISPERSION option (deviance, Pearson); default as set in MODEL
PROBABILITY = *scalar*	Probability level for confidence limits; default 0.95
OMODEL = *pointer*	Pointer to settings of options of the current MODEL statement, given unit labels corresponding to the option names of MODEL (starting with 'distribution')
PMODEL = *pointer*	Pointer to settings of parameters of the current MODEL statement, given unit labels corresponding to the parameter names of MODEL (starting with 'y'), only refers to the first setting of Y, FITTEDVALUES and RESIDUAL
STATISTICS = *variates*	Saves all the statistics that could be displayed for the first Y variate by the 'summary' setting of the PRINT option of the fitting directives FIT, ADD etc
CIMETHOD = *string token*	Method to use to calculate confidence intervals for nonlinear models (exact, quadratic); default quad
IGNOREFAILURE = *string*	Whether to ignore failure to fit a generalized linear model (yes, no); default no
MAXIMALMODEL = *formula structure*	Saves the maximal model (as defined by TERMS)
FITMODEL = *formula structure*	Saves the currently-fitted model (including any contrast functions)
FITCONSTANT = *scalar*	Saves a scalar containing the value one if the constant is included in the fitted model, or zero otherwise
FITTYPE = *scalar*	Saves a scalar to indicate the type of model that has been fitted
SAVE = *identifier*	Specifies save structure of model; default * i.e. that from latest model fitted

Parameters

Y = *variates*	Response variates for which results are to be saved; default is the list of response variates in the most recent MODEL statement
RESIDUALS = *variates*	Residuals for each Y variate, as specified by the RMETHOD option
FITTEDVALUES = *variates*	Fitted values for each Y variate
LEVERAGES = *variate*	Leverages of the units for each Y variate
ESTIMATES = *variates*	Estimates of parameters for each Y variate
SE = *variates*	Standard errors of the estimates

INVERSE = *symmetric matrix*	Inverse matrix from a linear or generalized linear model, inverse of second derivative matrix from a nonlinear model
VCOVARIANCE = *symmetric matrix*	Variance-covariance matrix of the estimates
DEVIANCE = *scalars*	Residual ss or deviance
DF = *scalar*	Residual degrees of freedom
TERMS = *pointer* or *formula structure*	Fitted terms (excluding constant)
ITERATIVEWEIGHTS = *variate*	Iterative weights from a generalized linear model
LINEARPREDICTOR = *variate*	Linear predictor from a generalized linear model
YADJUSTED = *variate*	Adjusted response of a generalized linear model
EXIT = *scalar*	Exit status from a generalized linear or nonlinear model
GRADIENTS = *pointer*	Derivatives of fitted values with respect to parameters in a nonlinear model
GRID = *variate*	Grid of function or deviance values from a nonlinear model
DESIGNMATRIX = *matrix*	Design matrix whose columns are explanatory variates and dummy variates
PEARSONCHISQUARE = *scalar*	Pearson chi-square statistic from a generalized linear model
STERMS = *pointer*	Saves the identifiers of the variates that have been smoothed in the current model
SCOMPONENTS = *pointer*	Saves a pointer to variates holding the nonlinear components of the variates that have been smoothed
NOBSERVATIONS = *scalar*	Number of units used in regression, excluding missing data and zero weights and taking account of restrictions
SEFITTEDVALUES = *variate*	Saves standard errors of the fitted values
SELINEARPREDICTOR = *variate*	Saves standard errors of the linear predictor
INFLATION = *variate*	Saves the variance inflation factors of the parameter estimates
UPPER = *variates*	Saves upper confidence limits for the parameter estimates
LOWER = *variates*	Saves lower confidence limits for the parameter estimates
MEANDEVIANCE = *scalars*	Saves the residual mean deviance (or mean square)
TDEVIANCE = *scalars*	Saves the total deviance (or sum of squares)
TDF = *scalars*	Saves the total degrees of freedom (corrected for the mean or uncorrected as displayed by the fitting directives)
TMEANDEVIANCE = *scalars*	Saves the total mean deviance (or mean square)
SUMMARY = *pointer*	Saves the summary analysis-of-variance (or deviance) table as a pointer with a variate or text for each column (source, d.f. etc)
ACCUMULATED = *pointer*	Saves the accumulated analysis-of-variance (or deviance) table as a pointer with a variate or text for each column (source, d.f. etc)
STATISTICS = *variates*	Saves all the statistics that could be displayed for the Y variate by the 'summary' setting of the PRINT option of the fitting directives FIT, ADD etc

Description

RKEEP allows you to copy information from a regression analysis (performed, for example, by a FIT, FITCURVE or FITNONLINEAR statement) into Genstat data structures. You do not need to declare the structures in advance; Genstat will declare them automatically to be of the correct type and length.

The Y parameter specifies the response variates for which the results are to be saved. Unusually for the first parameter of a directive, this has a default: if you leave it out, Genstat assumes that results are to be saved for all the response variates, as given in the previous MODEL statement.

The RESIDUALS, FITTEDVALUES, LEVERAGES, SEFITTEDVALUES and SELINEARPREDICTOR parameters allow you to save the standardized residuals, the fitted values, the leverages, the standard errors of the fitted values and the standard errors of the linear predictor. For example, RESIDUALS=R puts the residuals in a variate R. The RMETHOD option controls the type of residuals that are formed. You cannot save these values if you have set RMETHOD=* in the MODEL statement. The standard errors of fitted values are defined by:

s.e. = √(leverage × variance function × dispersion / weight)

where the variance function is calculated from the fitted value according to the setting of the DISTRIBUTION option of the current MODEL statement, and the dispersion is the fixed or estimated value of dispersion, as controlled by the DISPERSION and DMETHOD options of the MODEL and RKEEP directives.

The ESTIMATES and SE parameters save the parameter estimates and their standard errors; RKEEP puts them in variates, using the same order as in the display produced by the PRINT option of the directive used to fit the model. Alternatively, if you have used TERMS to define a maximal model, you can set option EXPAND=yes to reorder the estimates to their order in the maximal model (including missing values for the parameters not currently in the model). The variates saving these values are set up with labels; thus, you can refer to individual values in expressions using the labels as displayed when the estimates are printed. For example, to get the estimate of the constant into a scalar, you could put:

```
RKEEP ESTIMATES=Esti
SCALAR Const
CALCULATE Const = Esti$['Constant']
```

The UPPER and LOWER parameters allow you to save upper and lower confidence limits for the parameter estimates. The probability for the confidence interval is specifed by the PROBABILITY option, with default 0.95. The CIMETHOD option controls the method used with nonlinear models. The default setting, quadratic, uses the same method as for other types of regression, basing the limits on a quadratic surface fitted to the likelihood surface around the optimum. These may be poor approximations if the surface is very non symmetric. The alternative setting, exact, calculates the limits directly from the likelihood surface.

The INFLATION parameter allows the variance inflation factors of the parameters to be saved.

The INVERSE parameter allows you to save the inverse matrix as a symmetric matrix: that is, $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ where $\boldsymbol{X}$ is the design matrix. This matrix is the same for all response variates.

The VCOVARIANCE parameter saves the variance-covariance matrix of the estimates for each response variate: these are formed by multiplying the inverse matrix by the relevant variance estimate based on the estimated dispersion, or on the dispersion that you have supplied.

The DEVIANCE parameter lets you save the residual sum of squares, or the *deviance* for distributions other than Normal. The DF parameter saves the residual degrees of freedom, and the MEANDEVIANCE parameter saves the residual mean deviance. The TDEVIANCE parameter saves the total deviance, the TDF parameter saves the total degrees of freedom (corrected for the mean or uncorrected as displayed by the fitting directives), and the TMEANDEVIANCE parameter saves the total mean deviance.

The LINEARPREDICTOR parameter lets you save the linear predictor of a generalized linear

model; the values of the linear predictor are the same as the fitted values if the link function is the identity function.

The ITERATIVEWEIGHTS parameter saves a variate containing the iterative weights used in the last cycle of the iteration for fitting a generalized linear model. The iterative weights do not contain any contribution from the weights that can be specified, whether or not the model is iterative, by the WEIGHTS option of the MODEL directive, and they are 1.0 for ordinary linear regression.

The YADJUSTED parameter saves the adjusted response variate used in the last cycle of the iteration for fitting a generalized linear model; with the identity link function this is the same as the response variate.

The Pearson chi-square statistic can be saved using the PEARSONCHI parameter of RKEEP. It is calculated as the sum of the squared Pearson residuals. This can be used as an alternative to the deviance for testing goodness of fit; see Nelder & McCullagh (1989).

The EXIT parameter of RKEEP provides a code that indicates the success or type of failure of an iterative fit. Codes 0-7 are relevant to standard curves and general nonlinear models, and codes 0 and 8-13 are for generalized linear models:

0 Successful fitting
1 Limit on number of cycles has been reached without convergence
2 Parameter out of bounds
3 Likelihood appears constant
4 Failure to progress towards solution
5 Some standard errors are not available because the information matrix is nearly singular
6 Calculated likelihood may be incorrect because of missing fitted values
7 Curve is close to a limiting form
8 Data incompatible with model
9 Predicted mean or linear predictor out of range
10 Invalid calculation for calculated link or distribution
11 All units have been excluded from the analysis
12 Iterative process has diverged
13 Failure due to lack of space or data access
14 Function returned a missing value

With a generalized linear model, unless you set option IGNOREFAILURE=yes, the EXIT code is the only information that you can save if the fit has been unsuccessful. Alternatively, with a nonlinear model or when IGNOREFAILURE=yes, RKEEP will save any information that may be available. (You may thus, for example, be able to discover more about the cause of the failure.)

The derivatives of the fitted values with respect to each parameter in a standard curve or general nonlinear model can be stored in variates using the GRADIENTS parameter. You can use these quantities to assess the relative influence of each observation on a parameter; you can also construct a measure of leverage by summing the gradients for all the parameters.

The GRID parameter can be used to store a grid of values of the deviance (or any general function) following FITNONLINEAR.

The DESIGNMATRIX parameter allows you to save the matrix $\boldsymbol{X}$. The columns correspond to the parameters of the model, ordered as for the ESTIMATES parameter. For simple linear regression with a constant this has only two columns, the first containing ones and the second containing the values of the explanatory variate. Columns corresponding to aliased parameters are omitted, but you can use the corresponding option of TERMS to construct the full design matrix.

The PEARSONCHI parameter provides the Pearson chi-square statistic for dispersion, which is the same as the residual sum of squares for the Normal distribution, but is different to the deviance for other distributions. The STERMS and SCOMPONENTS parameters are relevant to generalized additive models. The STERMS parameter can be used to store a pointer to those

variates whose effects in the model are smoothed. The SCOMPONENTS parameter stores a pointer to variates, one for each smoothed variate in the same order as in STERMS, containing the fitted nonlinear component of each smoothed variate – this does not include the linear component or the constant term.

The NOBSERVATIONS parameter allows you to save the number of units used in the analysis, omitting units with missing values or excluded by restrictions. This will be the same as the total number of degrees of freedom plus one, except in a regression with no constant term and no explanatory factors when it will equal the total number of degrees of freedom.

The SUMMARY parameter can be used to save the summary analysis-of-variance (or deviance) table for each response variate. The summary table is saved as a pointer with a variate or text for each of its columns (source, d.f. etc). Similarly, the ACCUMULATED parameter can save the accumulated analysis-of-variance (or deviance) tables.

The STATISTICS parameter saves all the statistics that could be displayed for each response variate by the 'summary' setting of the PRINT option of the fitting directives FIT, ADD etc. Alternatively, the STATISTICS option can be used to save the statistics for the first response variate specified by the MODEL statement.

The DISPERSION option allows you to define the value to be used for the dispersion parameter when calculating the standard errors. The DMETHOD option indicates how this should be calculated if DISPERSION is not set. By default the deviance is used but you can set DMETHOD=Pearson to request the Pearson chi-square statistic to be used instead.

Options OMODEL and PMODEL allow you to save pointers containing information about the current model. The labels of the pointers can be specified in either lower or upper case, or any mixture. OMODEL can be set to a pointer to store information about each of the options set in the previous MODEL statement. For example, the statement

```
RKEEP [OMODEL=Om]
```

will allow you to refer to the current variate of weights (if one was set in the WEIGHTS option of MODEL) as Om['weights']. Whether or not a variate was set, the statement

```
MODEL [WEIGHTS=Om['weights']] Newobs
```

will allow a new analysis with the same weighting as the old.

The pointer Om has 16 values, with suffixes corresponding to the options of MODEL in the defined order. Similarly, the statement

```
RKEEP [PMODEL=Pm]
```

will set up a pointer storing the (eight) current parameter settings of the previous MODEL statement. However, if there was more than one response variate, the first value of the pointer will be the identifier of the first response variate only: the others are not stored. Similarly, only the fitted-values and residuals variates for the first response will be pointed at. For example, the identifier Pm[1] or Pm['y'] can be used to refer to the current response variate after the RKEEP statement above.

The MAXIMALMODEL option saves the maximal model (as defined by TERMS). The FITMODEL option saves the model that has currently been fitted, including any contrast functions (i.e. POL, REG, COMPARISON, SSPLINE or LOESS). The FITCONSTANT option saves a scalar containing the value one if the constant is included in the fitted model, or zero otherwise. The FITTYPE option saves a scalar to indicate the type of model that has been fitted: 1 for an ordinary regression or generalized linear model (FIT), 2 for a generalized nonlinear model (FIT with the CALCULATION option set), 3 for a standard curve (FITCURVE) and 4 for a nonlinear model (FITNONLINEAR).

Options: EXPAND, DISPERSION, RMETHOD, DMETHOD, PROBABILITY, OMODEL, PMODEL, STATISTICS, CIMETHOD, IGNOREFAILURE, MAXIMALMODEL, FITMODEL, FITCONSTANT, FITTYPE, SAVE.

Parameters: Y, RESIDUALS, FITTEDVALUES, LEVERAGES, ESTIMATES, SE, INVERSE, VCOVARIANCE, DEVIANCE, DF, TERMS, ITERATIVEWEIGHTS, LINEARPREDICTOR, YADJUSTED, EXIT, GRADIENTS, GRID, DESIGNMATRIX, PEARSONCHISQUARE, STERMS, SCOMPONENTS, NOBSERVATIONS, SEFITTEDVALUES, SELINEARPREDICTOR, INFLATION, UPPER, LOWER, MEANDEVIANCE, TDEVIANCE, TDF, TMEANDEVIANCE, SUMMARY, ACCUMULATED, STATISTICS.

Reference

McCullagh, P. & Nelder, J.A. (1989). *Generalized Linear Models* (second edition). Chapman and Hall, London.

See also

Directives: FIT, FITCURVE, FITNONLINEAR, RKESTIMATES.
Genstat Reference Manual 1 Summary section on: Regression analysis.

RKESTIMATES

Saves estimates and other information about individual terms in a regression analysis.

Options

FACTORIAL = *scalar*	Limit on number of factors and variates in a model term; default 3
Y = *variate*	Response variate for which results are to be saved; default is the last response variate in the save structure
SAVE = *identifier*	Provides the regression save structure for the analysis from which the estimates are to be saved; default * takes the save structure from the most recent regression

Parameters

TERMS = *formula*	Model terms for which information is required
ESTIMATES = *tables* or *scalars*	Table or scalar to store the estimated regression coefficients for each term
SE = *tables* or *scalars*	Table or scalar to store the standard errors of the estimated regression coefficients
VCOVARIANCE = *symmetric matrices*	Symmetric matrix or scalar to store the variances and covariances between the estimates of each term
DF = *scalars*	Number of degrees of freedom for each term
POSITIONS = *tables* or *scalars*	Positions of the estimates in the variate of estimates as saved from RKEEP when option EXPAND=yes

Description

RKESTIMATES allows you to save estimates and other information about terms in a regression or generalized linear model analysis into Genstat data structures. You do not need to declare the structures in advance; Genstat will declare them automatically to be of the correct type and size.

By default the results are saved from the most recent analysis, that is for the last y-variate in the most recent MODEL statement. Alternatively, you can use the SAVE option to specify the save structure from another analysis (see the SAVE option of MODEL). Again, the default is to save the information for the last y-variate, but you can use the Y option to specify another one.

The TERMS parameter specifies a model formula, which Genstat expands to form the series of model terms about which you wish to save information. As in FIT, the FACTORIAL option sets a limit on the number of factors and variates in each term. Any term containing more than that limit is deleted. You can include the single-line text 'constant' (in any case) to refer to the constant term.

The subsequent parameters allow you to specify identifiers of data structures to store the various types of information for each of the terms that you have specified. The ESTIMATES parameter saves estimates for each term, in a table if the term involves factors or in a scalar if it involves only variates. Similarly the SE parameter saves standard errors for the estimates. The VCOVARIANCE parameter saves the variances and covariances between the estimates of each term, in a symmetric matrix if the term involves factors or in a scalar if it involves only variates. The DF parameter saves the number of degrees of freedom for the terms, in scalars. Finally, the POSITIONS parameter saves the positions where the estimates can be found in the variate of estimates that would be saved by the ESTIMATES parameter of RKEEP when its option EXPAND=yes. (This allows you, for example, to obtain correlations between the estimates of different terms out of the variance-covariance matrix that can be saved by the VCOVARIANCE parameter of RKEEP.)

Options: FACTORIAL, Y, SAVE.
Parameters: TERMS, ESTIMATES, SE, VCOVARIANCE, DF, POSITIONS.

See also
Directives: FIT, FITCURVE, FITNONLINEAR, RKEEP.
Genstat Reference Manual 1 Summary section on: Regression analysis.

ROTATE

Does a Procrustes rotation of one configuration of points to fit another.

Options

PRINT = *string tokens*	Printed output required (rotations, coordinates, residuals, sums); default * i.e. no printing
SCALING = *string token*	Whether or not isotropic scaling is allowed (yes, no); default no
STANDARDIZE = *string tokens*	Whether to centre the configurations (at the origin), and/or to normalize them (to unit sum of squares) prior to rotation (centre, normalize); default cent,norm
SUPPRESSREFLECTION = *string token*	Whether to suppress reflection (yes, no); default no

Parameters

XINPUT = *matrices*	Inputs the fixed configuration
YINPUT = *matrices*	Inputs the configuration to be fitted
XOUTPUT = *matrices*	To store the (standardized) fixed configuration
YOUTPUT = *matrices*	To store the fitted configuration
ROTATION = *matrices*	To store the rotation matrix
RESIDUALS = *matrices* or *variates*	To store distances between the (standardized) fixed and fitted configurations
RSS = *scalars*	To store the residual sum of squares

Description

The ROTATE directive provides orthogonal Procrustes rotation. You must set the parameters XINPUT and YINPUT, which specify respectively the fixed configuration and the configuration that you want to be translated and rotated; these are called X and Y above. The other parameters are used for saving results from the analysis. For X and Y to refer to the same set of objects they must have the same number of rows, and each object must be represented by the same row in both X and Y. If the XINPUT matrix is $n \times p$ and the YINPUT matrix is $n \times q$, Genstat does the analysis using matrices that are $n \times r$, where r is $\max(p, q)$. The smaller matrix is expanded with columns of zeros, as explained above.

The PRINT option specifies which results you want to print; the settings are as follows.

coordinates	specifies that the fixed and fitted configurations are to be printed; note that the fixed configuration is printed after any standardization (see below), and the fitted configuration is printed after standardization and rotation.
residuals	prints the residual distances of the points in the fixed configuration from the fitted points; this is after any standardization and rotation.
rotations	prints the orthogonal rotation matrix.
sums	prints an analysis of variance giving the sums of squares of each configuration, and the residual sum of squares; if scaling is used, the scaling factor is also printed.

The three other options of the ROTATE directive control the form of analysis. The SCALING option specifies whether you want least-squares scaling to be applied to the standardized YINPUT matrix when finding the best fit to the fixed configuration. You should set SCALING=yes if you want scaling; Genstat will then print the least-squares scaling factor with the analysis of variance. By default there is no scaling.

The STANDARDIZE option specifies what preliminary standardization is to be applied to the

XINPUT and YINPUT matrices. It has settings:

centre	centre the matrices to have zero column means;
normalize	normalize the matrices to unit sums of squares.

The default is STANDARDIZE=centre,normalize. The initial centring ensures that the configurations are translated to have a common centroid, and thus automatically provides the best translation of *Y* to match *X*. The normalization arranges that the residual sum of squares from rotating *X* to *Y* is the same as that for rotating *Y* to *X*. Switching off both centring and standardization is rarely advisable, but can be requested by putting STANDARDIZE=*.

With some methods of multivariate analysis, for example the analysis of skew-symmetry, the direction of travel about the origin is important. It is then undesirable to perform a reflection as part of the rotation: the SUPPRESSREFLECTION option can be used to prevent this. The default setting is no, which allows reflection to take place.

Options: PRINT, SCALING, STANDARDIZE, SUPPRESSREFLECTION.
Parameters: XINPUT, YINPUT, XOUTPUT, YOUTPUT, ROTATION, RESIDUALS, RSS.

See also
Procedures: GENPROCRUSTES, PCOPROCRUSTES, SAGRAPES.
Genstat Reference Manual 1 Summary section on: Multivariate and cluster analysis.

SCALAR

Declares one or more scalar data structures.

Options

VALUE = *scalar*	Value for all the scalars; default is a missing value
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used to identify the scalars in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the scalars
VALUE = *scalars*	Value for each scalar
DECIMALS = *scalars*	Number of decimal places for printing
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for the contents of each structure
MAXIMUM = *scalars*	Maximum value for the contents of each structure
DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represents a date and time

Description

A scalar data structure stores a single number. The IDENTIFIER parameter lists the identifiers of the scalars that are to be declared.

Values can be assigned to the scalars by either the VALUE option or the VALUE parameter. The option defines a common value for all the structures in the declaration, while the parameter allows them each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence. However, if you do not define a value explicitly for a scalar, Genstat gives it a missing value.

The DECIMALS parameter allows you to define a number of decimal places to be used by default when each symmetric matrix is printed. You can associate a text of extra annotation with each scalar using the EXTRA parameter. The MINIMUM and MAXIMUM parameters allow you to define lower and upper limits on the values in each symmetric matrix. Genstat then prints warnings if any values outside that range are allocated to the scalar. The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each scalar to indicate that the scalar stores a date and time, and to define a format to be used for this, by default, when it is printed; details are given in the description of the PRINT directive.

If the MODIFY option is set to yes any existing attributes and values of the scalars are retained; otherwise these are lost. The IPRINT option can be set to specify how the scalars will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself), while the ANOVA directive will print the identifier and the extra text for each y-variate.

Options: VALUE, MODIFY, IPRINT.

Parameters: IDENTIFIER, VALUE, DECIMALS, EXTRA, MINIMUM, MAXIMUM, DREPRESENTATION.

See also
Directive: VARIATE.
Genstat Reference Manual 1 Summary section on: Data structures.

SET

Sets details of the "environment" of a Genstat job.

Options

INPRINT = *string tokens*	Printing of input as in PRINT option of INPUT (statements, macros, procedures, unchanged); default unch
OUTPRINT = *string tokens*	Additions to output as in PRINT option of OUTPUT (dots, page, unchanged); default unch
DIAGNOSTIC = *string tokens*	Defines the least serious class of Genstat diagnostic which should still be generated (messages, warnings, faults, extra, unchanged); default unch
ERRORS = *scalar*	Number of errors that a job may contain before it is abandoned (0 implies no limit); default is to leave unchanged
FAULT = *text*	Sets the Genstat fault indicator (for example, FAULT=* clears the last fault); default is to leave the indicator unchanged
PAUSE = *scalar*	Number of lines to output before pausing (interactive use only; 0 implies no pausing); default is no change
PROMPT = *text*	Characters to be printed for the input prompt; default is to leave unchanged
NEWLINE = *string token*	How to treat a new line (significant, ignored); default is no change
CASE = *string token*	Whether lower- and upper-case (small and capital) letters are to be regarded as identical in identifiers (significant, ignored); default is no change
FIELDWIDTH = *scalar*	Fieldwidth to be used as a default minimum by PRINT and other output commands
SIGNIFICANTFIGURES = *scalar*	Minimum number of significant figures to be supplied in the default formats determined by PRINT and other output commands
SEEDS = *pointer* or *scalar*	Defines the current default seeds to be used for random numbers in various parts of Genstat
RUN = *string token*	Whether or not the run is interactive (interactive, batch); by default the current setting is left unchanged
UNITS = *identifier*	To (re)set the current units structure; default is to leave unchanged
BLOCKSTRUCTURE = *identifier*	To (re)set the internal record of the most recent BLOCKSTRUCTURE statement; default is to leave unchanged
TREATMENTSTRUCTURE = *identifier*	To (re)set the internal record of the most recent TREATMENTSTRUCTURE statement; default is to leave unchanged
COVARIATE = *identifier*	To (re)set the internal record of the most recent COVARIATE statement; default is to leave unchanged
ASAVE = *identifier*	To (re)set the current ANOVA save structure; default is to leave unchanged
MSAVE = *identifier*	To (re)set the current save structure for multivariate analysis; default is to leave unchanged

DSAVE = *identifier*	To (re)set the current save structure for the high-resolution graphics environment; default is to leave unchanged
RSAVE = *identifier*	To (re)set the current regression save structure; default is to leave unchanged
TSAVE = *identifier*	To (re)set the current time-series save structure; default is to leave unchanged
VSAVE = *identifier*	To (re)set the current REML save structure; default is to leave unchanged
VCOMPONENTS = *identifier*	To (re)set the current REML model definitions, as specified by VCOMPONENTS and VSTRUCTURE; default is to leave unchanged
WORDLENGTH = *string token*	Length of word (8 or 32 characters) to check in identifiers, directives, options, parameters and procedures (long, short); default * i.e. no change
CAPTIONS = *string tokens*	Controls which captions are displayed (minor, major, meta, unchanged); default unch
TYPESET = *string tokens*	Controls when typesetting commands within textual strings are used (output, graphics); if unset, the existing setting is left unchanged
CMETHOD = *string token*	Controls whether number settings for colour options and parameters are interpreted as RGB values or as numbers of standard colours (rgb, standard); if unset, the existing setting is left unchanged
DATASPACE = *scalar* or *variate*	Updates the current data space allocations; if unset, the existing allocations are left unchanged
WORKINGDIRECTORY = *text*	Sets the working directory; default is to leave this unchanged
ALGORITHMS = *string token*	Controls the use of enhanced computing algorithms (standard, mkl); if unset, the existing setting is left unchanged
ACTIONAFTERFAULT = *string token*	Controls what happens after a fault (continue, stop); if unset, the existing setting is left unchanged
UNSETDUMMY = *string token*	Controls what happens if you specify an unset dummy as the setting of an option or parameter that expects another type of data structure (fault, ignore, warn); if unset, the existing setting is left unchanged
LANGUAGE = *text*	Text with either one or two values to specify a preferred language for output and (optionally) a second choice in case the preferred language is unavailable
YEAR2DIGITBREAK = *scalar*	Controls how 2 digits can be used to specify years
†TIMEWITHSECONDS = *string token*	Controls whether seconds are included with the time12 and time24 date representations; (absent, present, unchanged); default unch

No parameters

Description

The default of SET is to do nothing: that is, each option by default leaves the corresponding attribute of the environment unchanged. Of course you have to start somewhere, so an initial

environment is defined at the start of any Genstat program; the corresponding initial settings of the options of SET, known as the *initial defaults*, are described below.

The INPRINT option controls what parts of a Genstat job supplied in the current input channel are recorded in the current output file; the input channel can be either an input file or the keyboard. Three parts are distinguished: explicit statements; statements, or parts of statements, that you have supplied in macros using either the ## notation or the EXECUTE directive; and statements that you have supplied in procedures. The initial default is to record nothing if the output is to the screen, otherwise to record the statements. This aspect of the environment can be modified also by the PRINT option of the INPUT directive and by the INPRINT option of JOB.

The OUTPRINT option controls how the output from many Genstat directives starts: the output can be preceded by a move to the top of a new page, or by a line of dots beginning with the line number of the statement producing the analysis, or by both. If output is directly to the screen, no new pages are given. The initial default is to give neither if output is to the screen, otherwise to give a new page and a line of dots. Alternatively, this aspect can be modified by the PRINT option of the OUTPUT directive or by the OUTPRINT option of JOB. The lines of dots are produced by the directives for regression analysis, analysis of designed experiments, REML analysis, multivariate analysis and time series; also from the FLRV, FSSPM and SVD directives. If you give an analysis statement within a FOR loop, the line number preceding the line of dots is that of the ENDFOR statement rather than of the analysis statement. New pages are produced with any of the above, and with the GRAPH, HISTOGRAM and CONTOUR directives.

The DIAGNOSTIC option lets you control the level of diagnostic reporting. You might want to do this within a procedure, to prevent faults being reported to a user who does not need to know in detail what is going on inside the procedure. By initial default, all diagnostics – messages, warnings and faults – are printed. You can switch off messages by setting DIAGNOSTIC=warning, or switch off both messages and warnings by setting DIAGNOSTIC=fault. If you set DIAGNOSTIC=*, then no diagnostics will appear. The extra setting gives you extra information, in the form of a dump of the current state of the job; but this is likely to be useful only for developers of Genstat. Printing of diagnostics can also be controlled by the DIAGNOSTIC option of JOB.

The ERRORS option controls what Genstat does when many faults happen within a single job while in batch mode. By initial default, up to five errors per job are reported, and successive faults will not generate diagnostic messages. This ensures, for example, that input intended to be read by a READ statement will not generate many lines of diagnostics if execution halts because of a fault before the READ statement. Note, however, that this option does not affect the detailed error messages printed by the READ directive itself: these are controlled separately by the corresponding ERRORS option of READ. In interactive mode, the count of errors is restarted after each successful statement is issued, though the option is unlikely to be useful in this mode.

The FAULT option is provided primarily to allow procedure writers to modify the internal record that is kept of the most recent fault indicator. Setting FAULT=* clears the record; you can then use the GET directive to ascertain whether a fault has occurred since the record was cleared. You can also set the fault indicator to a particular diagnostic, for example

```
SET [FAULT='VA4']
```

A subsequent DISPLAY statement will then report the chosen fault in the standard way. The fault indicator is automatically cleared at the start of each job.

The PAUSE option lets you specify how many lines of output are produced at a time; you might, for example, want to read the output on a terminal screen before more output replaces it. Obviously this is relevant only in interactive mode, and may not be needed in the implementations of Genstat that provide a scrollable output window. By initial default, all output is sent to the current output channel as soon as it is available. Some computers can store the output, irrespective of whether Genstat itself has a scrollable window, and let you scroll forward and back to read it at leisure: others just provide keys to freeze the output while you are reading

a section, and then to continue to the next segment of output. If you set PAUSE=*n*, then after every *n* lines of output Genstat gives a prompt:

```
*Press RETURN to continue*
```

After you have read the displayed section of output, you can press the <RETURN> key to get the next *n* lines. The counting of lines is restarted each time you give a statement from the keyboard: it is not restarted between separate statements in a macro, procedure or auxiliary input channel. If you have specified that Genstat should echo input lines, these are included among the *n*. Once all the output has been displayed, Genstat prompts for further statements.

The PROMPT option specifies the characters used to prompt for interactive input. The initial default is the greater-than character followed by a space "> ". The prompt can also be modified by the PROMPT option of JOB. Other prompts are used by READ and EDIT, and these cannot be altered.

The NEWLINE option allows you to cancel the initial default whereby a new line (<RETURN>) is a terminator both for strings within a string list (1.6.2) and for a statement (1.8). Thus, for example, if you specify

```
SET [NEWLINE=ignored]
```

you need no longer use a backslash (\) to continue a statement onto a new line, since <RETURN> is no longer interpreted as the end of a statement. But you will then have to terminate each statement explicitly with a colon.

The CASE option specifies whether upper-case and lower-case letters are to be treated as the same in identifiers. The initial default is that upper and lower case are not the same; thus, an identifier X is distinct from an identifier x. If CASE is set to ignored, then in later statements, both x and X are treated as the same identifier, X. Thus the structure with identifier x cannot be referenced, unless CASE is later reset to significant.

The FIELDWIDTH option allows you to control the minimum fieldwidth that is used as a default by PRINT and other output commands. The initial default is 12.

In PRINT the default number of decimal places for a numerical structure is determined by calculating the number that would be required to print its mean absolute value to at least *d* significant figures. The initial default for *d* is four, but you can redefine this using the SIGNIFICANTFIGURES option.

The SEEDS option specifies the default seeds to be used to generate random numbers in various areas of Genstat. You can set SEED to a scalar to define a single seed to be used for all the areas. Alternatively, you can supply a pointer to define a different seed for each area. The elements of the pointer should be labelled to indicate the area concerned: for example 'calculate', and 'randomize' for random-number functions and the RANDOMIZE directive respectively. The easiest way to see the possibilities is to save the current seeds using the SEEDS option of the GET directive; this saves a pointer with elements labelled automatically. You will notice, though, that the GET pointer represents each seed as a variate (with several values) rather than a scalar. This is because, once any randomization has been done in an area, there is too much seed information to store in a single number. Variates are equally valid for the elements of the SET pointer. So you can save the current seeds using GET, and then restore them by using the same pointer in SET.

The RUN option controls whether Genstat interprets the program as being in batch or in interactive mode; this assumed mode is independent of whether the program really is being run in batch or interactively. Initially, a program is taken to be in interactive mode only if the first input channel and the first output channel are both connected to a terminal. The setting of the assumed mode has two effects – on recovery from faults, and on how EDIT operates.

The UNITS option provides another way of setting the *units structure* in addition to the UNITS directive. The setting can be the identifier of a variate or text structure; this will become the default labelling structure of other variates, texts or factors with the same length, in those

directives that use such labels. The setting can also be a scalar to specify the default number of units. The setting of the UNITS option is lost at the end of each job within a program.

The BLOCKSTRUCTURE, TREATMENTSTRUCTURE, COVARIATE, ASAVE, DSAVE, MSAVE, RSAVE, TSAVE, VSAVE and VCOMPONENTS options specify special *save structures* for graphical and analysis directives. You can set the options only to an identifier that you have previously established by the SPECIAL option of the GET directive or by the SAVE options of the various analysis directives themselves. For example, if two sets of regression analyses are in progress in one job, the SET directive can be used to switch between them:

```
MODEL [SAVE=S1] Y1
FIT X1
MODEL [SAVE=S2] Y2
FIT X1
SET [RSAVE=S1]
FIT X1,X2
```

This program fits the regression of Y1 on X1, using save structure S1, then the regression of Y2 on X1 with save structure S2. Finally, it fits the regression of Y1 on X1 and X2, because the current regression save structure is changed to S1 before the last FIT statement. The settings of these options are all lost at the end of a job.

The WORDLENGTH option controls the number of characters that are stored and checked in identifiers and names of directives, procedures, options, parameters and functions. In releases prior to 4.2 this was always eight, but from 4.2 onwards you can choose between eight (WORDLENGTH=short) and 32 (WORDLENGTH=long). This can also be controlled by the JOB directive and, within a procedure, by the PROCEDURE directive. The default is to leave the setting unchanged.

The CAPTIONS option controls which captions are displayed by directives and procedures. This can be used inside a procedure to suppress irrelevant captions that would be produced by the procedures or directives that it calls. The setting can be restored by the RESTORE option of the PROCEDURE statement, or by saving the current setting using GET, and then restoring it by using another SET. The initial default is to display all types of caption.

The TYPESET option controls whether typesetting commands within textual strings (see PRINT) are recognised used in output and in labels and titles on graphs. The initial default is to use them in both.

The CMETHOD option is useful if you have programs from Release 10 or earlier that use the old way of specifying graphics colours. Prior to Release 11, you had to use one of Genstat's 256 standard colours, and redefine its RGB definition, if necessary, using the COLOUR directive. In Release 11, the representation of colours was changed to allow you to use standard colour names (see PEN for details). So virtually all options and parameters of the directives and library procedures that define colours were modified to take strings or texts as their settings. Further flexibility was given by interpreting numeric settings directly as RGB values. However, if you have a program from Release 10 or earlier that relies on the old standard colours, you can put

```
SET [CMETHOD=standard]
```

to interpret numeric settings of colour options and parameters later in your program as standard colour numbers instead of RGB values.

The DATASPACE option allows you to increase the current data space allocations. You can set this to a variate of length three to specify a different size for each of the three types of data: real numbers (for numeric data), integers (for factors and system information) and characters (for texts). Alternatively, you can set it to a scalar to specify the same size for all three types. The sizes are measured in blocks of 32768 values. If any of the data spaces is already larger than the specified size, its size is left unchanged. This option can be useful if you know that your next analysis is likely to require lot of space – it is more efficient to reserve all the space at once, rather than leaving it to Genstat to expand each allocation every time that it becomes full.

The WORKINGDIRECTORY option allows you to set the working directory (the default directory where Genstat will open or save files).

The ALGORITHMS option allows you to request the use of enhanced computing algorithms. The initial default, at the start of any Genstat run, is to use only the standard algorithms. However, if you set ALGORITHMS=mkl, it will use algorithms from the Intel® Math Kernel Library for operations such as eigenvalue decompositions and matrix inversion. These should provide much faster performance with large problems.

The ACTIONAFTERFAULT option allows you to control what happens if a fault occurs inside a procedure or during a batch run. The initial default, at the start of any Genstat run, is that execution of the procedure or the batch script stops. However, you can set ACTIONAFTERFAULT to continue to request that it continues instead. The FAULT option of GET can be used to access the most recent fault code, so that you can make your own decision about what to do next if a fault occurs.

A dummy is a data structure that stores the identifier of a data structure. This can be useful with options and parameters that expect another type of data structure. If you supply a dummy, it will be replaced by the identifier that it stores. The UNSETDUMMY option controls what happens if the dummy is unset. The initial default is to give a warning, and replace the dummy by the default for the option or parameter if one has been defined, or otherwise to treat the option or parameter as though it had not been set. If you set UNSETDUMMY to ignore, no warning is given. Finally, if you set it to fault, unset dummies are treated as faults.

Some Genstat commands can now provide output in languages other than English. The LANGUAGE option allows you to supply a text with either one or two values to specify your preferred language in its first value, and (optionally) your second choice in its second value. Output will then be generated in your preferred language if that is available. Otherwise it may be in your second-choice language or, if neither are available, the command will generate the ordinary English output.

The YAR2DIGITBREAK option controls how two digits can be used to specify years: whether thse represent years in the 1900's or the 2000's. YAR2DIGITBREAK specifies the cut-off date: dates less than this value represent years beginning 20, and two digit dates greater than or equal to this value will represent years beginning 19. For example, if it is set to 30, years in the range 00 - 29 will represent the years 2000 - 2029 and years in the range 30 - 99 represent the years 1930 - 1999. Alternatively, the value 0 ensures that all 2 digit dates belong to the 1900's, while the value 100 means that they all belong to the 2000's.

The TIMEWITHSECONDS option controls whether seconds will be present or absent in output with the time12 and time24 date representations. The default is to leave the current setting unchanged.

Options: INPRINT, OUTPRINT, DIAGNOSTIC, ERRORS, FAULT, PAUSE, PROMPT, NEWLINE, CASE, FIELDWIDTH, SIGNIFICANTFIGURES, SEEDS, RUN, UNITS, BLOCKSTRUCTURE, TREATMENTSTRUCTURE, COVARIATE, ASAVE, DSAVE, MSAVE, RSAVE, TSAVE, VSAVE, VCOMPONENTS, WORDLENGTH, CAPTIONS, TYPESET, CMETHOD, DATASPACE, WORKINGDIRECTORY, ALGORITHMS, ACTIONAFTERFAULT, UNSETDUMMY, LANGUAGE, YEAR2DIGITBREAK.
Parameters: none.

See also
Directives: GET, PROCEDURE.
Genstat Reference Manual 1 Summary section on: Program control.

SETALLOCATIONS

Runs through all ways of allocating a set of objects to subsets.

Options

NREQUIRED = *scalar*	Number of allocations that are required; default 1
UNIQUE = *string token*	Whether only unique allocations are to be formed, allowing the reordering of the subsets (yes, no); default no
NFOUND = *scalar*	Number of allocations that has been found
NPOSSIBLE = *scalar*	Saves the total of allocations that can be formed
GROUPS = *factor* or *pointer*	Saves the allocations, in a single factor if NREQUIRED = 1, otherwise in a pointer to NFOUND factors
UNITS = *variate*	Supplies numbers for the objects; if unset, the positive integers 1, 2 ... are used
START = *factor*	Previous allocation; if unset the allocations start as a partitioning of the objects in the ordering in the UNITS variate

Parameters

SETSIZE = *scalars*	Number of objects in each subset
ELEMENTS = *variates* or *pointers*	Saves the objects allocated to each subset, in a single variate if NREQUIRED = 1, otherwise in a pointer to NFOUND variates

Description

The SETALLOCATIONS directive allows you to form all the ways in which a set of objects can be allocated to subsets. For example, suppose we have 4 objects numbered 1 ... 4 to allocate to two subsets of size 2. There are 6 possible ways of forming the allocations: {1, 2 : 3, 4}, {1, 3 : 2, 4}, {1, 4 : 2, 3}, {2, 3 : 1, 4}, {2, 4 : 1, 3} and {3, 4 : 1, 2}. If, however, the ordering of the subsets is unimportant, there are only three. For example {1, 2 : 3, 4} is then the same as {3, 4 : 1, 2}.

The sizes of the subsets are specified by the SETSIZE parameter, and the UNIQUE option can be set to yes to indicate that their ordering is unimportant (so only *unique* allocations are then formed). The NREQUIRED option indicates how many allocations you want to form (default 1). If you set NREQUIRED to a scalar containing a missing value, SETALLOCATIONS will save as many allocations as it can find.

You can use the NPOSSIBLE option to find out how many allocations are possible. The NFOUND option is useful if you request more allocations than are possible – it indicates how many allocations SETALLOCATIONS has actually been able to find.

The GROUPS option saves the allocations in factors (each with a level for each subset). If NREQUIRED = 1, it saves a single factor. Alternatively, if NREQUIRED is greater than one, it saves a pointer containing NFOUND factors.

As an alternative, the ELEMENTS parameter allows you to save the allocations in variates. For example

```
SETALLOCATIONS 2,2; ELEMENTS=Sub1, Sub2
```

saves the first subset in variate Sub1 and the second in variate Sub2. Just one allocation has been formed here as the NREQUIRED option has default 1. If several allocations are formed, ELEMENTS saves them in pointers to variates. For example

```
SETALLOCATIONS [NREQUIRED=3] 2,2; ELEMENTS=Sub1, Sub2
```

saves three allocations: (Sub1[1] : Sub2[1]), (Sub1[2] : Sub2[2]), and (Sub1[3] : Sub2[3]). By default, the variates will contain the positive integers 1, 2 upwards, but you can

supply a variate containing others using the UNITS option.

By default, the first allocation is a partitioning of the objects in the same ordering as in the UNITS variate (or numerical order if UNITS is not set). However, if you want to run through the allocations in order, you can save the current allocation using the GROUPS option, and then use this as the setting of the START option to get the next one. For example, the following program runs through all the allocations of seven objects into subsets of size 2, 3 and 2.

```
SETALLOCATIONS   [NPOSSIBLE=Nposs; GROUPS=Alloc] 2,3,2
CALCULATE        Ntimes = Nposs - 1
FOR [INDEX=i; NTIMES=Ntimes]
  DUPLICATE      Alloc; NEWSTRUCTURE=Start
  SETALLOCATIONS [NREQUIRED=1; GROUPS=Alloc; START=Start]\
                 2,3,2
  " use allocation Alloc "
ENDFOR
```

Options: NREQUIRED, UNIQUE, NFOUND, NPOSSIBLE, GROUPS, UNITS, START.
Parameters: SETSIZE, ELEMENTS.

Action with RESTRICT
Not relevant.

See also
Directives: SETCALCULATE, SETRELATE.
Procedure: SUBSET.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

SETCALCULATE

Performs Boolean set calculations on the contents of vectors or pointers.

Options

`NULL` = *scalar*	Returns either 1 or 0 according to whether or not the result is a null (i.e. empty) set
`FREPRESENTATION` = *string token*	How to represent factors in a calculation that contains only factors (`levels`, `labels`); default `leve`
`TOLERANCE` = *scalar*	Tolerance to use when comparing numerical values; default 10^{-6}
`SUBSTITUTE` = *string token*	Whether to substitute dummies within pointers in the expression (`yes`, `no`); default `no`
`NOMESSAGE` = *string tokens*	Which warning messages to suppress (`novalues`); default `*` i.e. none

Parameter

expression	Expression defining the calculation to be performed

Description

The `SETCALCULATE` directive allows you to perform set operations. The calculation must have a single assignment, setting a pointer, variate, text or factor to the result of a set calculation involving other compatible structures. Calculations on pointers must involve only pointers, those on variates and those on texts can involve factors, while those on factors can involve either variates or texts but not both.

For example, you can form a variate `All` that contains all the distinct values that occur in either of a pair of variates `x` and `y` using the `.OR.` operator

```
SETCALCULATE all = x .OR. y
```

or all the (distinct) values that occur in both of them using the `.AND.` operator

```
SETCALCULATE both = x .AND. y
```

The available operators are as follows:

`.OR.`	represents the Boolean "or" operation: for example `x.OR.y` produces a vector that contains any item that is in either `x` or `y`
`.AND.`	represents the Boolean "and" operation: for example `x.AND.y` produces a vector that contains any item that is in both `x` and `y`
`.EOR.`	represents "either or": for example `x.EOR.y` produces a vector that contains any item that is in either `x` or `y` but not both of them
`-`	represent "not", for example `x-y` produces a vectors that contains the items that are in `x` but not in `y`
`+`	synonym of `.OR.`
`,`	synonym of `.OR.`
`*`	synonym of `.AND.`

The `NULL` option can save a scalar whose value is 1 if the calculation generates an null set (i.e. one that has no members); otherwise the scalar is set to 0. The `FREPRESENTATION` option determines whether the values of factors are compared using their levels or their labels; by default the levels are used. The `TOLERANCE` option defines the tolerance to be used when comparing numbers. The default value 10^{-6} should be suitable, however, unless the variates contain very small values. If the calculation is operating on pointers, the `SUBSTITUTE` option

controls whether or not to replace any dummies that they contain by the structures to which they point. Finally, the NOMESSAGES option allows you to suppress the warning message that SETCALCULATE produces if one of the data structures in the calculation has no values.

Options: NULL, FREPRESENTATION, TOLERANCE, SUBSTITUTE, NOMESSAGES.
Parameter: unnamed.

Action with RESTRICT
SETCALCULATE ignores any restrictions on vectors in the expression.

See also
Directives: SETALLOCATIONS, SETRELATE, SET2FORMULA, CALCULATE.
Procedure: FDISTINCTFACTORS.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

SETOPTION

Sets or modifies defaults of options of Genstat directives or procedures.

Option

DIRECTIVE = *string token*	Directive (or procedure) to be modified

Parameters

NAME = *string tokens*	Option names
DEFAULT = *identifiers*	New default values

Description

The SETOPTION directive changes the default settings of options of a directive or procedure for the remainder of the current job. If you use this directive in your start-up file you can make the changed default apply in all your use of Genstat.

To achieve any effect, the option and both parameters of the directive must be set. The DIRECTIVE option specifies the name of the directive or procedure that is affected, and the NAME parameter indicates the option whose default is to be changed. The settings are strings, so need not be quoted because all directive and procedure names are valid as unquoted strings. The DEFAULT parameter is then set to a data structure to provide the new default that you want to be assumed. For example, the following statement modifies the PRINT option of the FIT directive.

```
SETOPTION [DIRECTIVE=FIT] PRINT; DEFAULT='deviance'
```

The usual default of the PRINT option in FIT is to print a statement of the model, a summary of the analysis, and the parameter estimates: this corresponds to the setting PRINT=model,summary,estimates. This SETOPTION statement therefore redefines the default so that any subsequent FIT statement in the job will report only the residual deviance unless you explicitly set the PRINT option.

The defined mode of the PRINT option of FIT is "strings" (8.1.2). However, the DEFAULT parameter of SETOPTION expects a data structure (to allow for all the other modes that might occur), and so it must be set to a text structure containing the string (or strings) that you want to be the default. Similarly, if the defined mode of the option is "numbers", "expression" or "formula", you must supply a variate, an expression structure or a formula structure containing the new default. If the defined mode is "identifier", the setting of DEFAULT is simply an identifier, which must be of the required type if this is specified in the definition of the directive or procedure.

To reset the PRINT option of FIT back to its usual default, you would need to give the statement

```
SETOPTION [DIRECTIVE=FIT] PRINT; DEFAULT=!t(model,summary,\
  estimates)
```

The SETOPTION directive can also be used to change defaults of any procedure: this may be a procedure in the standard Procedure Library, the Site Library or a personal library that you have already opened in the current program, or it may be a procedure that you have defined explicitly in the job.

Option: DIRECTIVE.
Parameters: NAME, DEFAULT.

See also

Directives: OPTION, PROCEDURE.
Genstat Reference Manual 1 Summary section on: Program control.

SETPARAMETER

Sets or modifies defaults of parameters of Genstat directives or procedures.

Option

DIRECTIVE = *string token*	Directive (or procedure) to be modified

Parameters

NAME = *string tokens*	Parameter names
DEFAULT = *identifiers*	New default values

Description

The SETPARAMETER directive changes the default settings of parameters of a directive or procedure for the remainder of the current job. If you use this directive in your start-up file you can make the changed default apply in all your use of Genstat. The option and parameters are used in exactly the same way as by the SETOPTION directive to change the defaults of options of directives and procedures. For full details see the description of SETOPTION.

Option: DIRECTIVE.
Parameters: NAME, DEFAULT.

See also

Directives: PARAMETER, PROCEDURE.
Genstat Reference Manual 1 Summary section on: Program control.

SETRELATE

Compares the distinct values contained in two data structures.

Options

FREPRESENTATION = *string token*	How to represent factors in a comparison between two factors (levels, labels, ordinals); default leve
LFACTORIAL = *scalar*	Limit on number of factors or variates in the terms formed from a LEFT formula; default * i.e. none
RFACTORIAL = *scalar*	Limit on number of factors or variates in the terms formed from a RIGHT formula; default * i.e. none
TOLERANCE = *scalar*	Tolerance to use when comparing numerical values; default 10^{-6}
SUBSTITUTE = *string token*	Whether to substitute dummies within LEFT or RIGHT pointers and formulae (yes, no); default no

Parameters

LEFT = *identifiers*	First structures in each comparison
RIGHT = *identifiers*	Second structures in each comparison
CONTAINS = *scalars*	Returns 1 or 0 according to whether or not LEFT contains RIGHT
EQUALS = *scalars*	Returns 1 or 0 according to whether or LEFT and RIGHT contain exactly the same distinct set of items
INCLUDEDIN = *scalars*	Returns 1 or 0 according to whether or not LEFT is included in RIGHT
DISTINCT = *scalars*	Returns 1 or 0 according to whether or not LEFT and RIGHT are distinct

Description

SETRELATE can compare the distinct values of any numerical structure (scalar, variate, table, matrix, diagonal matrix or symmetric matrix) with another numerical structure or with a factor. It can compare a factor either with another factor, or with a variate or a text. It can compare a text with another text, or two pointers. Finally, it can compare two formulae.

The LEFT and RIGHT parameters specify the structures to compare. The other parameters provide the results of the comparison as scalars containing the values 0 or 1. CONTAINS is set to 1 if the LEFT structure contains every (distinct) value in the RIGHT structure. EQUALS returns 1 if the sets of distinct values in the LEFT and RIGHT structures are identical. INCLUDEDIN equals 1 if the RIGHT structure contains every (distinct) values in the LEFT structure. DISTINCT equals 1 if the LEFT and RIGHT structures are have a null intersection, i.e. they have no values in common.

When comparing two factors, the FREPRESENTATION option specifies whether to use levels, labels or ordinal values. (The ordinal values are formed representing the levels, in numerical order, by the numbers 1, 2 and so on.) By default levels are used.

When comparing pointers and formulae, the SUBSTITUTE option controls whether any dummies that they contain are substituted by the data stuctures to which they point, before the comparison is made. Note, if any of those data structures is a dummy, it too is replaced, and so on until we reach a data structure that is *not* a dummy. However, if the original dummy (or any of the dummies to which it points) is unset, the original dummy is retained.

The LFACTORIAL and RFACTORIAL options can be used to set a limit on number of factors or variates in the terms formed from a LEFT or RIGHT formula, respectively; by default there are no limits.

The TOLERANCE option defines the tolerance to be used when comparing numbers. The

default value 10^{-6} should be suitable, however, unless the numbers are very small.

Options: FREPRESENTATION, LFACTORIAL, RFACTORIAL, TOLERANCE, SUBSTITUTE.
Parameters: LEFT, RIGHT, CONTAINS, EQUALS, INCLUDEDIN, DISTINCT.

Action with RESTRICT
SETRELATE ignores any restrictions on LEFT or RIGHT vectors.

See also
Directives: SETALLOCATIONS, SETCALCULATE, SET2FORMULA, CALCULATE.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

SET2FORMULA

Forms a model formula using a set of structures supplied in a pointer.

Option

METHOD = *string token*	Relationship of the structures within the formula (combined, crossed, nested); default comb

Parameters

POINTER = *pointers*	Sets of structures to be used to form the formulae
FORMULA = *formula structures*	Formulae constructed from the sets

Description

SET2FORMULA forms a model formula using the contents (factors and/or variates) of a pointer. The pointer is specified by the POINTER parameter, and the formula is saved by the FORMULA parameter.

The METHOD option defines how the formula is constructed. With the combined setting, the formula has a single model term combining all the structures: for example

```
SET2FORMULA !p(A,B,C); FORMULA=Fcomb
```

sets up Fcomb as the formula

```
A.B.C
```

The crossed setting links the contents of the pointer using the operator *, so it would form the formula

```
A*B*C
```

The nested setting uses the operator /, so it would form

```
A/B/C
```

Option: METHOD.
Parameters: POINTER, FORMULA.

See also

Directives: FORMULA, FCLASSIFICATION, REFORMULATE, SETCALCULATE, SETRELATE.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

SHELLEXECUTE

Launch executables or open files in another application using their file extension, PC Windows only.

No options

Parameters

FILE = *text*	Name of the file to execute
STATUS = *scalar*	Indicates whether the execution of the file was successful (0) or not (1)
MESSAGE = *text*	Saves the error message associated with a failure to execute the file

Description

The SHELLEXECUTE directive can be used to launch executables or start an application associated with a given document extension without knowing the name of the associated application under Windows. For example, you could start a web browser by using the file extension .html.

The FILE parameter specifies the name of the file to execute as a text. This must contain the absolute or relative pathname, for example

```
'C:/Consult/Data/Info.html'
```

or

```
'../../Project/Main.html'
```

Note we have used a forward slash (/) above as the directory separator character. The the backwards slash (\) is the Genstat continuation character, and would need to be doubled up (\\) to avoid it being interpreted as a continuation.

The STATUS parameter saves a scalar indicating whether the file execution was successful (0) or not (1). If the file execution was unsuccessful you can save the associated error message in a text using the MESSAGE parameter.

Options: none.
Parameters: FILE, STATUS, MESSAGE.

See also

Directive: SUSPEND.

SKIP

Skips lines in input or output files.

Options

CHANNEL = *scalar*	Channel number of file; default current channel of the specified type
FILETYPE = *string token*	Type of the file concerned (input, output); default inpu
STYLE = *string token*	Style to use when skipping output (plaintext, formatted); default * uses the current style of the channel

Parameter

identifiers	How many lines to skip; for input files, a text means skip until the contents of the text have been found, further input is then taken from the following line

Description

SKIP can be used with either input or output files. The FILETYPE and CHANNEL options indicate which file is to be skipped. By default this is the current input channel.

For input files you can skip over unwanted lines, which might be comments describing the data that is to follow, or might be some statements that you do not want to use in your current job. You can skip a specified number of lines, *n* say, by setting the parameter to a scalar containing the value *n*. Alternatively, you can skip everything up to and including a particular string of characters by setting the parameter to a text containing that string. For example,

```
SKIP [CHANNEL=2] 'Section 2'
```

will skip the contents of the input file on channel 2 from the current position until the string Section 2 is found. The next line to be read from channel 2 will then be the one immediately after the line containing Section 2.

For output files you can use SKIP to print blank lines to separate one section of output from another. You might want to do this if you had set the PRINT option SQUASH=yes to suppress the automatic blank lines within a section of output. For example,

```
PRINT [CHANNEL=2; IPRINT=*; SQUASH=yes] Heading
SKIP  [CHANNEL=2; FILETYPE=output] 2
PRINT [CHANNEL=2; IPRINT=*; SQUASH=yes] Table
```

places two blank lines between Heading and Table when printing their values to channel 2.

For an output file that has been opened in a style other than plain text (see OPEN), you can use the STYLE option to control whether the skipping is done in formatted or plain-text styles. If STYLE is not set, the default is to use the current style (as controlled by the OUTPUT directive).

Options: CHANNEL, FILETYPE, STYLE.
Parameter: unnamed.

See also

Directives: PAGE, READ, PRINT, CAPTION.
Genstat Reference Manual 1 Summary section on: Input and output.

SORT

Sorts units of vectors according to an index vector.

Options

INDEX = *vectors*	Variates, texts or factors whose values are to define the ordering; default is to use the first vector in the OLDVECTOR list
DIRECTION = *string token*	Order in which to sort (ascending, descending); default asce
DECIMALS = *scalar*	Number of decimal places to which to round before sorting numbers; default * i.e. no rounding

Parameters

OLDVECTOR = *vectors* or *pointers*	Factors, pointers, texts or variates whose values are to be sorted
NEWVECTOR = *vectors* or *pointers*	Structure to receive each set of sorted values; if any are omitted, the values are placed in the corresponding OLDVECTOR

Description

The SORT directive allows you to reorder the units of a list of vectors or pointers according to one or more "index" vectors. These can be specified explicitly using the INDEX option (and they need not be among the vectors actually sorted). If you omit the INDEX option, Genstat uses the first vector in the OLDVECTOR list. The DECIMALS option allows you to define the number of decimal places that are taken into account for an index variate: for example DECIMALS=0 would round each value to the nearest integer. If you do not set this, there is no rounding. The DIRECTION option controls whether the ordering is into ascending or descending order; by default DIRECTION=ascending.

The vectors or pointers whose values are to be sorted are listed by the OLDVECTOR parameter. The units of each structure are permuted in exactly the same way, into an ordering determined from the index vectors. The NEWVECTOR parameter allows you to specify new vectors to contain the sorted values, and thus keep the unsorted values in the original vectors. For example

```
SORT [INDEX=Name] Age,Income,Name,Sex; NEWVECTOR=A,*,N,S
```

would place the sorted values of Age, Name and Sex into A, N and S; as there is a null entry (*) corresponding to Income in the NEWVECTOR list, the sorted incomes would replace the original values of Income. Any undeclared vector in the NEWVECTOR list is declared implicitly to match the corresponding OLDVECTOR.

Options: INDEX, DIRECTION, DECIMALS.
Parameters: OLDVECTOR, NEWVECTOR.

Action with RESTRICT

You can restrict the index vector, or any of the oldvectors, to sort only a subset of the units. Each of the units that is not in the subset is left in its original position.

See also

Directive: CALCULATE.
Function: SORT
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

SPLOAD

Loads Genstat spreadsheet files.

Options

†PRINT = *string token*	What to print (catalogue, directory, summary); default cata
SCOPE = *string token*	When SPLOAD is used within a procedure, this allows the data structures to be created in program that called the procedure (SCOPE=external) or in the main program itself (SCOPE=global) rather than within the procedure (local, external, global); default loca
REDEFINE = *string token*	Whether to allow existing structures to have their type redefined (no, yes); default no
SYSTEM = *string token*	Whether to include Genstat system structures in the catalogue (yes, no); default no
UNNAMED = *string token*	Whether to include unnamed structures in the catalogue (yes, no); default no
TEMPMISSING = *string token*	Whether to read temporarily missing values as missing (yes, no); default no

Parameters

FILENAME = *texts*	Names of spreadsheet files
SHEETNAME = *texts*, *variates* or *scalars*	Names or numbers of the sheets to read from each file; default * reads them all
ISAVE = *pointers*	Stores the identifiers of the structures loaded from each file

Description

The SPLOAD directive can be used to load data from a Genstat (i.e. GSH) spreadsheet file, specified using the FILENAME parameter. If the file is a multi-paged spreadsheet, the SHEETNAME parameter can be used to specify which sheet (or page) to read. If SHEETNAME is not set, all the pages are read. By default, a summary is produced listing the data that have been read; this can be suppressed by setting option PRINT=*. The SYSTEM and UNNAMED options control whether to include system and unnamed structures in the summary.

The PRINT option controls printed output, with the following settings:

catalogue	lists the contents of the file (default),
directory	includes the directory details with the name of the file in the catalogue, and
summary	prints a summary of the values in each data structure in the file.

By default, unnamed structures are excluded from the catalogue, but you can set the UNNAMED option to yes to include them.

The SCOPE option is useful when SPLOAD is being used within a procedure. By default, the structures are created within the procedure. Alternatively, you can set SCOPE=external, to request that they are created in the program that called the procedure (which may itself be a procedure). Or you can set SCOPE=global to create them in the main program itself. However, care needs to be taken to ensure that there is no conflict with any existing structures.

If SPLOAD reads a structure from the spreadsheet file that has the same name as an existing structure, it will overwrite the values and attributes of the existing one, so long as the type is the same. Otherwise a VA 8 diagnostic message will be generated and SPLOAD will fail. To force

SPLOAD to change the type of existing structures you can set the option REDEFINE=yes.

The TEMPMISSING option controls the input of temporarily missing values. These are values that have been set to missing temporarily in the spreadsheet, and for which the original (non-missing) values are still available. The default is to read the original values, but you can set TEMPMISSING=yes to read them as missing values instead.

The ISAVE parameter can be set to a pointer to store the identifiers (i.e. column names) read from the file. The pointer can then be used to refer to the loaded data (within a procedure, for example).

Options: PRINT, SCOPE, REDEFINE, SYSTEM, UNNAMED, TEMPMISSING.
Parameters: FILENAME, SHEETNAME, ISAVE.

See also
Directive: READ.
Procedures: FILEREAD, GRIBIMPORT, IMPORT, EXPORT, SPCOMBINE.
Genstat Reference Manual 1 Summary section on: Input and output.

SSPM

Declares one or more SSPM data structures.

Options

`TERMS` = *formula*	Terms for which sums of squares and products are to be calculated; default `*`
`FACTORIAL` = *scalar*	Maximum number of vectors in a term; default 3
`FULL` = *string token*	Full factor parameterization (`yes, no`); default `no`
`GROUPS` = *factor*	Groups for within-group SSPMs; default `*`
`DF` = *scalar*	Number of degrees of freedom for sums of squares; default `*`

Parameters

`IDENTIFIER` = *identifiers*	Identifiers of the SSPMs
`SSP` = *symmetric matrices*	Symmetric matrix to contain the sums of squares and products for each SSPM
`MEANS` = *variates*	Variate to contain the means for each SSPM
`NUNITS` = *scalars*	Number of units or sum of weights for each SSPM
`WMEANS` = *pointers*	Pointers to variates of group means for each SSPM

Description

The SSPM structure stores a matrix of corrected sums of squares and products, and associated information, as used for regression and some multivariate analyses. You can form values for SSPM structures by the `FSSPM` directive. However, most multivariate and regression analyses can be done without declaring and forming an SSPM explicitly.

An SSPM comprises four structures (identified by their suffixes). Their labels can be specified in either upper or lower case, or any mixture.

`[1]` or `['Sums']` is a symmetric matrix containing the sums of squares and products. The number of rows and columns of this matrix will equal the number of parameters defined by the expanded terms list: that is, the number of variates plus the number of dummy variates generated by the model formula. (See the `TERMS` directive.)

`[2]` or `['Means']` is a variate containing the mean for each variate or dummy variate.

`[3]` or `['Nunits']` is a scalar holding the total number of units used in constructing the sums of squares and products matrix. If the SSPM is weighted, this scalar will hold the sum of the weights.

A within-group SSPM has one additional element:

`[4]` or `['Wmeans']` is a pointer, pointing to variates holding within-group means. There is one variate for each row of the `'Sums'` matrix plus one extra. They are all of the same length, namely the number of levels of the `GROUPS` factor. The extra variate holds counts of the number of units in each group.

The `TERMS` option of the `SSPM` directive defines the model for whose components the sums of squares and products are to be calculated. In the simplest case the model is just a list of variates, but you can use more complex model formulae, involving variates and factors; this is done in conjunction with the `FACTORIAL` and `FULL` options.

You can form a within-group matrix of sums of squares and products by specifying the relevant factor with the `GROUPS` option.

Sometimes you may already have calculated values for the matrix of sums of squares and products. You can then assign them to the component structures of the SSPM for example by `READ`. You would still, however, need to set the number of degrees of freedom associated with the matrix, and for that you use the `DF` option.

The parameter lists let you specify identifiers for the four components of an SSPM. You can

have declared them previously (and you can have given them values), but if so they must be of the correct type.

Options: TERMS, FACTORIAL, FULL, GROUPS, DF.
Parameters: IDENTIFIER, SSP, MEANS, NUNITS, WMEANS.

See also
Directives: FSSPM, CVA, FCA, PCO, PCP, TERMS, FORMULA, SCALAR, SYMMETRICMATRIX, VARIATE.
Procedures: FCORRELATION, FVCOVARIANCE, ROBSSPM.
Genstat Reference Manual 1 Summary sections on: Data structures, Multivariate and cluster analysis.

STEP

Selects terms to include in or exclude from a linear, generalized linear or generalized additive model according to the ratio of residual mean squares.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, changes, confidence); default mode,summ,esti,chan
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
INRATIO = *scalar*	Criterion for inclusion of terms; default 1.0
OUTRATIO = *scalar*	Criterion for exclusion of terms; default 1.0
MAXCYCLE = *scalar*	Limit on number of times to repeat stepwise selection, unless no change is made; default 1
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95

Parameter

formula	List of explanatory variates and factors, or model formula

Description

STEP modifies the current regression model, which may be linear, generalized linear or generalized additive, in order to achieve the biggest "improvement". Terms in the specified formula are dropped from the current model if they are already there, or are added to it if they are not. For each term, the residual sum of squares (or deviance) and the residual degrees of freedom are recorded; then Genstat reverts to the original model before trying the next term.

The current model is finally modified by the best term, according to a criterion based on the variance (or deviance) ratios. In a linear model, suppose that the residual sum of squares and

residual degrees of freedom of the current model are s_0 and d_0, and of the model after making a one-term change are s_1 and d_1. If the variance ratio for any term that is dropped is less than the value of the setting of the OUTRATIO option, then the term that most reduces or least increases the residual mean square is dropped. That is, when the dispersion is being estimated, a term will be dropped only if at least one term has

$\{(s_1-s_0) / (d_1-d_0)\} / \{s_0/d_0\}$ < OUTRATIO

When the dispersion is fixed, the equation becomes

$\{(s_1-s_0) / (d_1-d_0)\}$ < OUTRATIO

If you have set OUTRATIO=*, then no term is dropped. Note that, though the criteria are ratios of variances, you should not interpret them as F-statistics with the usual interpretation of significance. The probability levels would need to be adjusted to take account of correlations between the explanatory variables concerned, and the number of changes being considered.

If no term satisfies the criterion for dropping, then the term that most reduces the residual mean square will be added to the model if its variance ratio is greater than the setting of the INRATIO option. That is, when the dispersion is being estimated, if

$\{(s_0-s_1) / (d_0-d_1)\} / \{s_1/d_1\}$ > INRATIO

When the dispersion is fixed, the equation becomes

$\{(s_0-s_1) / (d_0-d_1)\}$ > INRATIO

Likewise, if you have set INRATIO=*, no term will be added.

If neither criterion is met, the current model is left unchanged.

Usually, the effect of the STEP directive is to make one change of a stepwise regression search. You can make STEP do forward selection by setting the MAXCYCLE option to define a maximum number of changes; STEP will stop at this limit, or earlier if no further changes can be made.

The changes setting of the PRINT option produces a list of terms with the corresponding residual mean squares (or deviances) and residual degrees of freedom, ordered according to the sizes of the residual mean squares; this list is not available for display later by the RDISPLAY directive. The INRATIO and OUTRATIO options are explained above. The rest of the options are as in the FIT directive, except that there is no CONSTANT option.

Options: PRINT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION, INRATIO, OUTRATIO, MAXCYCLE, PROBABILITY.
Parameter: unnamed.

Action with RESTRICT

If a TERMS statement was given before fitting the model, any restrictions on the variates or factors in the model will have been implemented then. So any restrictions on vectors involved in the model specified by STEP will be ignored. If no TERMS statement has been given and STEP involves new terms not already in the model, restrictions on the variates or factors in these terms will be taken into account and may cause the units involved in the regression to be redefined.

See also

Directives: MODEL, TERMS, FIT, ADD, DROP, SWITCH, TRY.
Procedures: RSCREEN, RSEARCH.
Genstat Reference Manual 1 Summary section on: Regression analysis.

STOP

Ends a Genstat program.

No options or parameters

Description

The STOP directive indicates the end of a Genstat program, thus telling the computer that you have finished using Genstat. It also ends the existing job, so there is no need to give an ENDJOB statement beforehand. Any input that follows a STOP statement is ignored.

Options: none.
Parameters: none.

See also

Directives: JOB, ENDJOB.
Genstat Reference Manual 1 Summary section on: Program control.

STORE

To store structures in a subfile of a backing-store file.

Options

PRINT = *string token*	What to print (catalogue); default *
CHANNEL = *scalar*	Channel number of the backing-store file where the subfile is to be stored; default 0, i.e. the workfile
SUBFILE = *identifier*	Identifier of the subfile; default SUBFILE
LIST = *string token*	How to interpret the list of structures (inclusive, exclusive, all); default incl
METHOD = *string token*	How to append the subfile to the file (add, overwrite, replace, update); default add, i.e. clashes in subfile identifiers cause a fault (note: replace overwrites the complete file)
PASSWORD = *text*	Password to be stored with the file; default *
PROCEDURE = *string token*	Whether subfile contains procedures only (yes, no); default no
UNNAMED = *string token*	Whether to list unnamed structures (yes, no); default no
MERGE = *string token*	Whether or not to merge the structures with the existing contents of the subfile (yes, no); default no

Parameters

IDENTIFIER = *identifiers*	Identifiers of the structures to be stored
STOREDIDENTIFIER = *identifiers*	Identifier to be used for each structure when it is stored

Description

The STORE directive allows you to store data structures or procedures in a backing-store file. The file can be opened using the OPEN directive, and there is also a backing-store workfile attached to channel 0 which is automatically deleted at the end of a Genstat run.

Each backing-store file contains a number of *subfiles*. Each subfile starts with a *catalogue*, recording which structures it stores. Then come the attributes and the values of each data structure. A subfile name can be either an unsuffixed identifier or a suffixed identifier with a numerical suffix. The identifiers of subfiles are kept in a separate catalogue to the identifiers of data structures, so you do not need to worry about keeping the identifiers of data structures and subfiles distinct. However, if you use a suffixed identifier for a subfile, Sub[1] say, you cannot also use the identifier Sub. There are two types of subfiles. *Ordinary subfiles* can hold any type of structures except procedures; *procedure subfiles* hold only procedures (and their dependent structures).

Whenever you store a structure in a subfile, Genstat automatically stores also all the associated structures to which it points. If these associated structures also point to further structures, then they are stored too, and so on. Some of the structures may be unnamed and some structures may be system structures. For example

```
TEXT [VALUES=A,B,C] T
FACTOR [LABELS=T; VALUES=1...3] F
STORE F
```

creates a subfile containing factor F. The complete definition of factor F depends on text T to supply level names. So T is stored too. The text T depends on a system structure (indicating the length of each line), which is therefore also stored. Hence to save factor F, Genstat has actually saved three structures. However, this is all automatic, so you do not need to worry about any of the details of the system structures.

When you store a structure with a suffixed identifier, Genstat may have to set up a series of pointer structures if they are not already present. An example is:

```
VARIATE [VALUES=1,2] V[1,2]
STORE [PRINT=catalogue] V[1]
```

The first line sets up a pointer structure V, pointing to V[1] and V[2]. To store variate V[1], a pointer structure V has to be set up in the subfile, pointing to V[1] only. Thus two structures are saved on backing store, namely V and V[1]. The original pointer V in the program is left unchanged. (If the example had stored the whole of V, no such complications would have arisen.)

The structures to be stored are specified by the IDENTIFIER parameter. The CHANNEL option indicates the backing-store file to use, and the SUBFILE option specifies the subfile that is created. Both these options can be omitted; by default the file will be the workfile, and the subfile will be called SUBFILE. The structures that are stored in the subfile are merely copies of the structures in the job, so the original structures remain available for further use within the job.

The STOREDIDENTIFIER parameter allows you to give a structure a different name within the subfile: For example,

```
VARIATE [VALUES=10.2,15.3,21.4,16.8,22.3] Weight
STORE Weight; STOREDIDENTIFIER=WtWeek2
```

stores a structure with identifier Weight within Genstat as a structure with identifier WtWeek2 in the backing-store file. If you want to rename only some of the structures, you can either respecify the existing identifier, or insert * at the appropriate point in the list. For example, you could store X and Y, renaming only Y as Yy, by

```
STORE X,Y; STOREDIDENTIFIER=X,Yy
```

or by

```
STORE X,Y; STOREDIDENTIFIER=*,Yy
```

You can give an unnamed structure in the list of either parameter. For example

```
STORE !(10.2,15.3,21.4,16.8,22.3); STOREDIDENTIFIER=WtWeek2
```

But of course you will not be able to retrieve any structure that has been stored as an unnamed structure (except perhaps as a dependent structure of another structure).

All the structures in a subfile must have distinct identifiers, and Genstat will report a fault if you try to give two the same name. You thus need to be careful if you are storing structures inside a procedure, as the same identifier can be used for one structure within the procedure, and for another one outside; you cannot store both in the same subfile.

Procedures that have been retrieved automatically from libraries cannot be stored by STORE.

You can set option PRINT=catalogue to obtain a catalogue of the subfiles in the backing-store file, and of the structures in the subfile just created. If you also set option UNNAMED=yes Genstat will also list any unnamed structures, with details of how they depend on each other.

The LIST option controls how the IDENTIFIER list is interpreted. The default setting inclusive simply stores the structures that have been listed.

Alternatively, if you set LIST=all Genstat will store all the structures in the current job that have identifiers and whose types have been defined. If the statement is inside a procedure, then only the structures defined within the procedure are stored. If you are storing procedures, then this setting will store all procedures that you have created explicitly in this job, by PROCEDURE or RETRIEVE statements.

Finally, you can set LIST=exclusive to store everything that you have not included in the IDENTIFIER parameter: that is, all the other named structures that are currently accessible, or all the other procedures that have been created in this job. Note, though, that some of the structures in the IDENTIFIER list may be stored if they are needed to complete the set of structures to be stored. If you use this setting, the STOREDIDENTIFIER parameter is ignored. For example

```
TEXT [VALUES=a,b] T
FACTOR [LABELS=T] F
TEXT [VALUES='variate  text'] Vt
VARIATE V; EXTRA=Vt
```

creates four named structures, `T`, `F`, `V` and `Vt`. The statement

```
STORE [LIST=inclusive] T
```

stores the text `T`;

```
STORE [LIST=all]
```

stores all the four structures that have identifiers;

```
STORE [LIST=exclusive] F,T
```

stores `Vt` and `V`; and

```
STORE [LIST=exclusive] Vt,T
```

results in all four structures being saved, because `V` points to `Vt`, and `F` points to `T`.

If a subfile of the specified name already exists on the backing-store file, the storing operation will usually fail. You can then set option `METHOD=overwrite` to overwrite the old subfile, that is, to replace the old subfile with a new subfile; alternatively, you can put `METHOD=replace` to form a new backing-store file containing only the new subfile. Setting `METHOD=update` adds new structures to an existing subfile. The `MERGE` option then controls what happens if a data structure that is being added to the file is already present; by default it overwrites the previous version but, if you put `MERGE=yes`, only new structures are added to the file.

To make your files secure, you can specify a password using the `PASSWORD` option. Once you have done this, you must include the same password in any future use of `STORE` or `MERGE` with this same userfile; spaces, case and new lines are significant in the password. You cannot change the password in a userfile once you have set it, but you can use the `MERGE` directive to create a new userfile with no password or with a new password. If you set the password to be a text whose values have been restricted, the restriction is ignored.

The `PROCEDURE` option indicates whether the subfile is to store procedures (`PROCEDURE=yes`), or ordinary data structures.

Options: `PRINT`, `CHANNEL`, `SUBFILE`, `LIST`, `METHOD`, `PASSWORD`, `PROCEDURE`, `UNNAMED`, `MERGE`.
Parameters: `IDENTIFIER`, `STOREDIDENTIFIER`.

See also
Directives: `RETRIEVE`, `CATALOGUE`, `MERGE`, `OPEN`, `RECORD`, `RESUME`.
Procedures: `EXPORT`, `DBEXPORT`.
Genstat Reference Manual 1 Summary section on: Input and output.

STRUCTURE

Defines a compound data structure.

Options

NAME = *text*	Single-valued text defining a name for the type of structure, which must not clash with the name of any existing type of structure
STRUCTURELIST = *string token*	Whether or not the structure consists of a list (of any length) of structures of the same type or types (yes, no); default no

Parameters

LABEL = *texts*	Single-valued texts defining the labels of the elements of the structure
SUFFIX = *scalars*	Suffix numbers for the elements; default assumes the numbers 1, 2 ...
TYPE = *texts*	Texts defining the allowed types for each element
COMPATIBLE = *texts*	Defines aspects to check for compatibility with the first element

Description

The STRUCTURE directive allows you to define customized compound data structures for use, for example, in procedures. The NAME option supplies a single-valued text to define the name to be used for the new "type" of data structure. This can then be used as a setting for the TYPE parameter in either the OPTION or PARAMETER directives within a procedure, to indicate that the option or parameter concerned must be supplied with this type of structure. The case of the letters in the name is not significant. So they can be specified in capitals, or in lower case, or in any mixture.

The parameters of the directive define the contents of the structure. The LABEL parameter lists the labels to be used with each element of the structure, and the SUFFIX parameter lists the corresponding suffix numbers (by default the numbers 1, 2, etc.). The TYPE parameter allows you to define the types of structure that are allowed in each element (which may be any of the standard Genstat data structures, or other customized types), and the COMPATIBLE parameter allows you to define aspects that must be compatible with the first element of the structure similarly to the COMPATIBLE parameter of the OPTION and PARAMETER directives. These are checked when the structure is declared, and when it is used as an option or parameter setting of a procedure that requests that type.

For example, we could define a complex matrix structure by

```
STRUCTURE [NAME='complex_matrix'] 'real','imaginary'; \
  TYPE='matrix'; COMPATIBLE=!t(rows,columns)
```

A particular complex matrix, Cmat say, could then be declared using the DECLARE directive:

```
DECLARE [TYPE='complex_matrix'] Cmat
```

The elements of the compound structure can be referred to like those of an ordinary pointer declared using the POINTER directive with options CASE=ignored, ABBREVIATE=yes and FIXNVALUES=yes. So, the labels can be given in either upper or lower case or in any mixture, and each can be abbreviated to the minimum number of characters required to distinguish it from the previous labels. So the imaginary part of the complex matrix above could, for example, be referred to as Cmat['imaginary'] or Cmat['IMAGINARY'] or simply Cmat['i'].

Options: NAME, STRUCTURELIST.
Parameters: LABEL, SUFFIX, TYPE, COMPATIBLE.

See also
Directives: DECLARE, POINTER, LRV, SSPM, TSM.
Genstat Reference Manual 1 Summary section on: Data structures.

SUSPEND

Suspends execution of Genstat to carry out commands in the operating system; this directive may not be available on some computers.

Options

SYSTEM = *text*	Commands for the operating system; default: prompt for commands (interactive mode only)
CONTINUE = *string token*	Whether to continue execution of Genstat without waiting for commands to complete (yes, no); default no
MINIMIZE = *string token*	Whether to minimize the console window (yes, no); default no

No parameters

Description

If you run the command

```
SUSPEND
```

(with no options) in Genstat *for Windows*, a command window will open, in which you can enter commands in the usual way. You can return to Genstat by closing the window (e.g. by typing EXIT).

You can use the SYSTEM option to specify the commands to run in the operating system. By default, Genstat then pauses while the commands run, and continues after they finish. This provides a convenient way to run an external program. For example, it is used by the _CDCALL procedure, included with CDNBLOCKDESIGN, to run the CycDesigN engine. CDNBLOCKDESIGN (and other CDN procedures that use _CDCALL) use the OPEN directive to open a file to contain the data for the engine, and the PRINT directive (with the CHANNEL option set to the filename) to form its contents. _CDCALL uses TXCONSTRUCT to construct the command for SUSPEND. The CycDesigN engine writes its output to another file, which can be read afterwards by CDNBLOCKDESIGN (or the other CDN procedures).

You can set option CONTINUE=yes if the operating-system commands do not need to run before your subsequent Genstat commands. For example, you might simply want to post a message to say that your Genstat run is executing.

You can set option MINIMIZE=yes to minimize the console window in which the comands run.

Note, however, that SUSPEND may not be available in all implementations of Genstat.

Options: SYSTEM, CONTINUE, MINIMIZE.
Parameters: none.

See also

Directives: PASS, EXTERNAL.
Genstat Reference Manual 1 Summary section on: Program control.

SVD

Calculates singular value decompositions of matrices.

Option

PRINT = *string tokens*	Printed output required (`left, singular, right`); default * i.e. no printing

Parameters

INMATRIX = *matrices*	Matrices to be decomposed
LEFT = *matrices*	Left-hand matrix of each decomposition
SINGULAR = *diagonal matrices*	Singular values (middle) matrix
RIGHT = *matrices*	Right-hand matrix of each decomposition

Description

Suppose that we have a rectangular matrix A with m rows and n columns, and that p is the minimum of m and n. The singular value decomposition can be defined as

$$ {}_mA_n = {}_mU_p \; {}_pS_p \; {}_pV_n' $$

The diagonal matrix S contains the p singular values of A, ordered such that

$$ s_1 \geq s_2 \geq \ldots \geq s_p \geq 0 $$

The matrices U and V contain the left and right singular vectors of A, and are orthonormal:

$$ U'U = V'V = I_p $$

The smaller of U and V will be orthogonal. So, if A has more rows than columns, $m>n$, $p=n$ and $VV'=I_p$.

The least-squares approximation of rank r to A can be formed as

$$ A_r = U_r S_r V_r' $$

where U_r and V_r are the first r columns of U and V, and S_r contains the first r singular values of A (Eckart & Young 1936).

The INMATRIX parameter specifies the matrices to be decomposed. The algorithm uses Householder transformations to reduce A to bi-diagonal form, followed by a QR algorithm to find the singular values of the bi-diagonal matrix (Golub & Reinsch 1971). The other parameters allow you to save the component parts of the decomposition: LEFT, SINGULAR and RIGHT for U, S and V respectively.

The PRINT option allows you to print any of the components of the decomposition; by default, nothing is printed. If any of the matrices is to be printed, all p columns are shown, even if you are storing only the first r columns.

Genstat will decide how many columns and singular values r to store, and will store that number for any of the components that you specify. If none of the matrices in the LEFT, SINGULAR and RIGHT lists has been declared in advance, the full number of singular values ($r=p$) is stored; otherwise Genstat sets r to the maximum number of columns contained in any of the matrices. If $r<p$, the first r singular values will be saved, along with the corresponding columns of singular vectors.

One practical application of the singular value decomposition is to form generalized inverses of matrices. If you use the singular value decomposition you obtain the Moore-Penrose generalized inverse, sometimes called the *pseudo-inverse*, and this is the method used by the GINVERSE procedure.

Option: PRINT.
Parameters: INMATRIX, LEFT, SINGULAR, RIGHT.

References

Eckart, C. & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, **1**, 211-218.

Golub, G.H. & Reinsch, C. (1971). Singular value decomposition and least squares solutions. *Numerische Mathematik*, **14**, 403-420.

See also

Directives: `DIAGONALMATRIX`, `MATRIX`, `NAG`, `FLRV`, `QRD`.

Functions: `SVALUES`, `LSVECTORS`, `RSVECTORS`.

Genstat Reference Manual 1 Summary sections on: Calculations and manipulation, Multivariate and cluster analysis.

SWITCH

Adds terms to, or drops them from a linear, generalized linear, generalized additive or nonlinear model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, confidence); default mode,summ,esti
NONLINEAR = *string token*	How to treat nonlinear parameters between groups (common, separate, unchanged); default unch
CONSTANT = *string token*	How to treat the constant (estimate, omit, unchanged, ignore); default unch
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95
AOVDESCRIPTION = *text*	Description for line in accumulated analysis of variance (or deviance) table when POOL=yes

Parameter

formula	List of explanatory variates and factors, or model formula

Description

SWITCH modifies the current regression model, which may be linear, generalized linear, generalized additive, standard curve or nonlinear. Terms in the specified formula are dropped from the current model if they are already there, or are added to it if they are not. It is best to give a TERMS statement before investigating sequences of models using SWITCH, in order to define a common set of units for the models to be explored. If no model is fitted after the TERMS

statement, the current model is taken to be the null model.

If the current model contains a smoothed term (specified e.g. by SSPLINE or LOESS) which is included in the formula specified by the parameter of SWITCH with a different number of degrees of freedom (or with the smoothing parameter set), SWITCH will then refit the smoothed term.

The model fitted by SWITCH will include a constant term if the previous model included one, and will not include one if the previous model did not. You can, however, change this using the CONSTANT option.

The options of SWITCH are the same as those of the FIT directive, but with the extra NONLINEAR option which controls whether separate nonlinear parameters are fitted to different groups when fitting curves, as in FITCURVE.

Options: PRINT, NONLINEAR, CONSTANT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION, PROBABILITY, AOVDESCRIPTION.
Parameter: unnamed.

Action with RESTRICT

If a TERMS statement was given before fitting the model, any restrictions on the variates or factors in the model will have been implemented then. So any restrictions on vectors involved in the model specified by SWITCH will be ignored. If no TERMS statement has been given and SWITCH introduces new terms into the model, restrictions on the variates or factors in these terms will be taken into account and may cause the units involved in the regression to be redefined.

See also

Directives: MODEL, TERMS, FIT, FITCURVE, ADD, DROP, STEP, TRY.
Functions: COMPARISON, POL, REG, LOESS, SSPLINE.
Genstat Reference Manual 1 Summary section on: Regression analysis.

SYMMETRICMATRIX

Declares one or more symmetric matrix data structures.

Options

ROWS = *scalar*, *vector* or *pointer* or *text*
: Number of rows, or labels for rows (and columns); default *

VALUES = *numbers*
: Values for all the symmetric matrices; default *

MODIFY = *string token*
: Whether to modify (instead of redefining) existing structures (yes, no); default no

IPRINT = *string tokens*
: Information to be used by default to identify the symmetric matrices in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*
: Identifiers of the symmetric matrices

VALUES = *identifiers*
: Values for each symmetric matrix

DECIMALS = *scalars*
: Number of decimal places for printing

EXTRA = *texts*
: Extra text associated with each identifier

MINIMUM = *scalars*
: Minimum value for the contents of each structure

MAXIMUM = *scalars*
: Maximum value for the contents of each structure

DREPRESENTATION = *scalars* or *texts*
: Default format to use when the contents represent dates and times

Description

A symmetric square matrix is symmetric about its leading diagonal: that is, the value in column *i* of row *j* is the same as that in column *j* of row *i*. For example:

```
1  2  3
2  1  4
3  4  1
```

Symmetric matrices often occur in statistics. Suppose, for example, that we have *n* random variables $X_1 \ldots X_n$. Then the covariance of X_i with X_j is the same as the covariance of X_j with X_i. The covariance matrix of the random variables is therefore symmetric: the off-diagonal elements of the matrix are the covariances (and the diagonal elements are the variances).

Because of this symmetry, Genstat stores only the diagonal elements and those below it; this is called the *lower triangle*. So you must specify only these values, whether in the declaration by SSPM or in a READ statement. (As always, you give them in row order: so if there are *n* rows, then for the first you supply one value, for the second two, and so on.) Likewise, Genstat prints only the lower triangle in output, for example with PRINT.

The ROWS option defines both the number of rows and the number of columns. The simplest way of doing this is to use a scalar to define the number of rows and columns explicitly. Alternatively, you can set ROWS to a variate, text or pointer, whose length then defines the number of rows and whose values will then be used as labels, for example when the symmetric matrix is printed. Finally, if you specify a factor, the number of levels defines the number of rows and the labels if available, or otherwise the levels, are used for labelling.

Values can be assigned to the symmetric matrices by either the VALUES option or the VALUES parameter. The option defines a common value for all the matrices in the declaration, while the parameter allows them each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

If the MODIFY option is set to yes any existing attributes and values of the symmetric matrices are retained (if still appropriate); otherwise these are lost.

The DECIMALS parameter allows you to define a number of decimal places to be used by default when each symmetric matrix is printed. You can associate a text of extra annotation with each symmetric matrix using the EXTRA parameter. The MINIMUM and MAXIMUM parameters allow you to define lower and upper limits on the values in each symmetric matrix. Genstat then prints warnings if any values outside that range are allocated to the matrix. The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each symmetric matrix to indicate that the matrix stores dates and times, and to define a format to be used for these, by default, when they are printed; details are given in the description of the PRINT directive.

The IPRINT option can be set to specify how the symmetric matrices will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

Options: ROWS, VALUES, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUES, DECIMALS, EXTRA, MINIMUM, MAXIMUM, DREPRESENTATION.

See also
Directives: DIAGONALMATRIX, LRV, MATRIX, SSPM.
Genstat Reference Manual 1 Summary section on: Data structures.

SYNTAX

Obtains details of the syntax of a command and the source code of a procedure.

No options

Parameters

COMMAND = *texts*	Single-line texts specifying the commands
NOPTIONS = *scalars*	Number of options for each command
NPARAMETERS = *scalars*	Number of parameters for each command
NAME = *texts*	Names of the options, and then the parameters, of each command
MODE = *texts*	Modes of the options and parameters
NVALUES = *pointers*	Number of values allowed for the options and parameters
VALUES = *pointers*	Allowed values for the options and parameters
DEFAULT = *pointers*	Default values for the options and parameters
SET = *texts*	Whether the options and parameters must be set
DECLARED = *texts*	Whether the options and parameters must have been declared
TYPE = *pointers*	Allowed types for the options and parameters
COMPATIBLE = *pointers*	Aspects of the options and parameters that must be compatible with the first parameter
PRESENT = *texts*	Whether the options and parameters must have values
LIST = *texts*	Whether the options have more than one setting (not relevant for the parameters
INPUT = *texts*	Whether the options and parameters only supply input information
DEFINITION = *texts*	Saves statements to define the syntax
SOURCE = *texts*	Saves the source code of a procedure

Description

SYNTAX enables you to obtain details about the syntax of a command (i.e. a directive or a procedure). The name of the command must be supplied in a single-value text, using the COMMAND parameter. The NOPTIONS parameter gives its number of options, and the NPARAMETERS parameter gives the number of parameters.

The other parameters give details of the options and parameters. These correspond to the parameters of the OPTION and PARAMETER directives.

The NAMES parameter saves a text containing the names of the options (if any), followed by the names of any parameters.

The MODE parameter saves a text giving the modes of the options and parameters: whether their settings should be a number (v), or an identifier of a data structure (p), or a string (t), or an expression (e), or a formula (f). These codes are exactly the same as those that indicate the mode of the values to appear within the brackets containing an unnamed structure.

The NVALUES saves a pointer defining how many values the structures that are supplied for options and parameters of mode p may contain. The element of the pointer is a scalar there is only one possibility, and a variate if there are several.

The VALUES saves a pointer containing the allowed set of values that may have been defined for options and parameters with modes t and v. The element of the pointer will be a text for an option or parameter of mode t, and either a scalar or a variate for an option or parameter of mode v.

The DEFAULT parameter saves a pointer containing the default settings that may have been

defined for the options and parameters with modes `t` and `v`.

The `SET` parameter saves a text containing `'yes'` or `'no'` according to whether or not the options and parameters must be set.

The `DECLARED` parameter saves a text containing `'yes'` or `'no'` according to whether or not the options and parameters of mode `p` must be set to a data structure that has already been declared.

The `TYPE` parameter saves a pointer containing a text to indicate the allowed types of the structures to which each option and parameter of mode `p` can be set.

The `COMPATIBLE` parameter saves a pointer containing a texts to specify aspects of the options and parameters that must be compatible with the first parameter.

The `PRESENT` parameter saves a text containing `'yes'` or `'no'` according to whether or not the options and parameters must be set to a data structure that has values.

The `INPUT` parameter saves a text containing `'yes'` or `'no'` according to whether or not the options and parameters are be used only to provide input to the command.

The `DEFINITION` parameter can save statements, in a text, to define the syntax. These start with a `DEFINE` statement for a directive or a `PROCEDURE` statement for a procedure, then an `OPTION` statement to define any options, and a `PARAMETER` statement to define any parameters.

The `SOURCE` parameter can save the source code of a procedure. This can be useful if you have a library containing the procedure, but no longer have the original source file. Note, though, that the source that you save will not be identical to the original source. When procedures are defined within Genstat, their source code is processed to remove comments and extraneous spaces in order to save storage space.

Parameters: `COMMAND`, `NOPTIONS`, `NPARAMETERS`, `NAME`, `MODE`, `NVALUES`, `VALUES`, `DEFAULT`, `SET`, `DECLARED`, `TYPE`, `COMPATIBLE`, `PRESENT`, `LIST`, `INPUT`, `DEFINITION`, `SOURCE`.

See also

Directives: `COMMANDINFORMATION`, `OPTION`, `PROCEDURE`.
Procedures: `SPSYNTAX`.
Genstat Reference Manual 1 Summary section on: Program control.

TABLE

Declares one or more table data structures.

Options

CLASSIFICATION = *factors*	Factors classifying the tables; default *
MARGINS = *string token*	Whether to add margins (yes, no); default no
VALUES = *numbers*	Values for all the tables; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the tables in output (identifier, extra, associatedidentifier); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the tables
VALUES = *identifiers*	Values for each table
DECIMALS = *scalars*	Number of decimal places for printing
EXTRA = *texts*	Extra text associated with each identifier
UNKNOWN = *identifiers*	Identifier for scalar to hold summary of unclassified data associated with each table
MINIMUM = *scalars*	Minimum value for the contents of each structure
MAXIMUM = *scalars*	Maximum value for the contents of each structure
DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represent dates and times
DATAVARIATE = *variates*	Records the identifier of the variate whose summaries are in the table
SUMMARYTYPE = *string tokens*	Records the type of summary that the table contains (counts, totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default * i.e. not recorded
PERCENTQUANTILE = *scalars*	Records the percentage points for which quantiles have been formed; default * i.e. not recorded
%MARGIN = *pointers*	Records the factors defining the margin over which the table has been converted to percentages

Description

Tables are used to store numerical summaries of data that are classified into groups. With Genstat, the classification into groups is specified by a set of factors. The table contains an element, called a *cell*, for each combination of the levels of the factors that classify it.

Tables are declared using the TABLE directive. The CLASSIFICATION option specifies the factors classifying the table.

Values can be assigned to the tables by either the VALUES option or the VALUES parameter. The option defines a common value for all the tables in the declaration, while the parameter allows them each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

If the MODIFY option is set to yes any existing attributes and values of the tables are retained (if still appropriate); otherwise these are lost.

The DECIMALS parameter allows you to define a number of decimal places to be used by

default when each table is printed. You can associate a text of extra annotation with each table using the EXTRA parameter. The MINIMUM and MAXIMUM parameters allow you to define lower and upper limits on the values in each table. Genstat then prints warnings if values outside that range are allocated to the table. The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each table to indicate that the table stores dates and times, and to define a format to be used for these, by default, when they are printed; details are given in the description of the PRINT directive.

A table can also have *margins*. There is then a margin for each classifying factor; this contains some sort of summary over the levels of that factor. For example, if you have a table in which the cells contain totals of the observations, you would want the marginal cells to contain totals across the levels of the factor. You can define a table to have margins when you declare it, by setting the MARGINS option of the TABLE directive to yes. Alternatively you can add margins later by the MARGIN directive.

Tables also have an associated scalar which collects a summary of all the observations for which any of the classifying factors has a missing value; these observations cannot be assigned to any cell of the table itself. This scalar is known as the *unknown cell* of the table. It can be given an identifier, so that you can refer to it, using the UNKNOWN parameter of the TABLE directive.

The IPRINT option can be set to specify how the tables will be identified in output:

identifier	uses the identifier;
extra	uses the EXTRA text;
associatedidentifier	uses the "associated identifier", if available; this is the identifier of the data variate from which the summaries in the table have been formed.

If IPRINT is not set, the tables will be identified in whatever way is usual for the section of output concerned.

The attribute of the table that records its data variate is set automatically when a table of summaries is formed by the TABULATE directive. If you have formed the summaries in some other way, you can use the DATAVARIATE parameter to record the relevant variate yourself. The SUMMARYTYPE parameter can set an attribute recording the type of summary that the table contains, and the PERCENTQUANTILE parameter can set an attribute recording the corresponding percentage if the table contains quantiles. (These are alse set automatically for tables formed by TABULATE.) The %MARGIN parameter can be set to a pointer of factors defining the margin of the table over which it has been converted to percentages. The pointer may contain just a scalar if the percentages have been formed over the "grand" margin (e.g. the grand total or grand mean). See the PERCENT procedure (which will set this attribute automatically) for further details. If any of these parameters is not set, the default is to leave the corresponding attribute of the table unchanged. To clear the existing value of one of these attributes, you can put a missing value into the corresponding parameter setting. For example

```
TABLE Tab; DATAVARIATE=*; SUMMARYTYPE=*; PERCENTQUANTILE=*;\
      %MARGIN=*
```

clears all these attributes for the table Tab.

Options: CLASSIFICATION, MARGINS, VALUES, MODIFY, IPRINT.

Parameters: IDENTIFIER, VALUES, DECIMALS, EXTRA, UNKNOWN, MINIMUM, MAXIMUM, DREPRESENTATION, DATAVARIATE, SUMMARYTYPE, PERCENTQUANTILE, %MARGIN.

See also
Directives: FACTOR, TABULATE, MARGIN, COMBINE.
Procedures: DTABLE, MTABULATE, PERCENT, SVSTRATIFIED, SVTABULATE, TABMODE, TABINSERT, TABSORT, T%CONTROL.
Genstat Reference Manual 1 Summary section on: Data structures.

TABULATE

Forms summary tables of variate values.

Options

PRINT = *string tokens*	Printed output required (counts, totals, nobservations, means, minima, maxima, variances, quantiles, sds, skewness, kurtosis, semeans, seskewness, sekurtosis); default * i.e. no printing
CLASSIFICATION = *factors*	Factors classifying the tables; default * i.e. these are taken from the tables in the parameter lists
COUNTS = *table*	Saves a table counting the number of units with each factor combination; default *
SEQUENTIAL = *scalar*	Used for sequential formation of tables; a positive value indicates that formation is not yet complete (see READ); default *
MARGINS = *string token*	Whether the tables should be given margins if not already declared (yes, no); default no
IPRINT = *string token*	Whether to print the identifier of the table or the identifier of the (associated) variate that was used to form it (identifier, extra, associatedidentifier); default iden
WEIGHTS = *variate*	Weights to be used in the tabulations; default * indicates that all units have weight 1
PERCENTQUANTILES = *scalar* or *variate*	Percentage points for which quantiles are required; default 50 (i.e. median)
OWN = *scalar* or *variate*	Specifies option settings for the OWNTAB subroutine and indicates that this is to supply the data values instead of the variates in the DATA list; default *
OWNFACTORS = *factors*	Factors whose values are to be read by OWNTAB (must include the factors of the classification set); default *
OWNVARIATES = *variates*	Variates whose values are to be read by OWNTAB (must include the DATA variates); default *
INCHANNEL = *scalar*	Channel number of the file from which the OWNTAB subroutine is to read the data (previously opened by an OPEN statement)
INFILETYPE = *string token*	Type of the OWN data file (input, unformatted); default inpu

Parameters

DATA = *variates*	Data values to be tabulated
TOTALS = *tables*	Tables to contain totals
NOBSERVATIONS = *tables*	Tables containing the numbers of non-missing values in each cell
MEANS = *tables*	Tables of means
MINIMA = *tables*	Tables of minimum values in each cell
MAXIMA = *tables*	Tables of maximum values in each cell
VARIANCES = *tables*	Tables of cell variances
QUANTILES = *tables* or *pointers*	Table to contain quantiles at a single PERCENTQUANTILE or pointer of tables for several

	PERCENTQUANTILEs (not available for sequential or OWN tabulation)
SDS = *tables*	Tables of standard deviations
SKEWNESS = *tables*	Tables of skewness coefficients
KURTOSIS = *tables*	Tables of kurtosis coefficients
SEMEANS = *tables*	Tables of standard errors of means
SESKEWNESS = *tables*	Tables of standard errors of skewness coefficients
SEKURTOSIS = *tables*	Tables of standard errors of kurtosis coefficients

Description

TABULATE allows you to produce the various types of tabular summary listed in the settings of its PRINT option. The variates whose values are to be summarized are listed with the DATA parameter. If you want to save the summaries in tables, for manipulating or for printing later on, you should list identifiers of the tables in the appropriate parameter list: for example, you would save the totals in a table T by including T in the list for the TOTALS parameter. The other parameters similarly give the other kinds of summary: numbers of non-missing values, means, minima, maxima, variances, quantiles, standard deviations, skewness, kurtosis and (within-cell) standard errors of means, skewness or kurtosis.

If you specify less tables in the lists than the number of DATA variates, Genstat produces accumulated summaries. For example, with

```
TABULATE Sales2001,Costs2001,Sales2002,Costs2002;\
  TOTALS=Totalsales,Totalcosts
```

the TOTALS list is recycled. So Totalsales will correspond to Sales2001 and Sales2002, and accumulate the totals from both variates. Similarly Totalcost will contain the totals from the variates Costs2001 and Costs2002. To avoid confusion, however, you are not allowed to specify table lists with differing lengths.

The simplest quantile, and the one produced by default, is the median (50% quantile), but the PERCENTQUANTILE option allows you to request any percentage point (between 0 and 100, of course). Moreover, by specifying a variate as the setting for PERCENTQUANTILE, you can obtain several quantiles at the same time. However, if you then want to save the results the setting of the QUANTILE parameter must be a pointer with length equal to the required number of quantiles, instead of a single table.

If you merely want to print the summaries, you do not usually need to list any tables; you need only specify the PRINT option. The only exception to this is with sequential tabulation, described at the end of this subsection.

The CLASSIFICATION option defines the classifying factors for the tables. This need not be set if at least one of the tables has already been declared (but then all the declared tables must have the same classifying factors). The MARGINS option determines whether or not the tables will have margins, if none have already been declared (and those that have been declared must be either all with margins or all without margins).

In the tables that correspond to the parameters of TABULATE, missing values of the data variates are ignored. So the NOBSERVATIONS parameter and the nobservations setting of the PRINT option provide the numbers of non-missing units of the data variates for each factor combination. You can however obtain a count of the numbers of units that would have contributed to each group if no values had been missing: you use the COUNTS option if you want to save the table, or put PRINT=counts if you want to print it. If any of the factor values are missing Genstat ascribes the corresponding units to the *unknown* cell associated with the table (see the TABLE directive).

If there are no observations in one of the groups, the corresponding cell will be zero in a table of numbers of observations or counts; in a table of totals, means, minima, maxima, variances, standard deviations, skewness, kurtosis or standard errors of means, skewness or kurtosis the cell

will contain a missing value.

Weighted tables can be obtained by setting the WEIGHT option to a variate of weights. You can, in general, think of weights as a set of multipliers which are applied to the data before any operations are performed. Thus, for most aspects of weighted tabulation you can replace x by wx and 1 by w (that is, n by Σw) in the standard formulae; see the table below. This is not what happens in the case of variances, standard deviations (which are square roots of the variances) and quantiles, but it is true for the other functions (including counts).

	Unweighted	Weighted
Count	n	$\Sigma\, w$
Total	$\Sigma\, x$	$\Sigma\, wx$
Nobservations	n	$\Sigma\, w$ (x not missing)
Mean	$\Sigma\, x/n$	$\Sigma\, wx$ / $\Sigma\, w$
Minimum	Min(x)	Min(wx)
Maximum	Max(x)	Max(wx)
Variance	$\Sigma(x - (\Sigma x/n))^2 / n-1$	$\Sigma w(x - (\Sigma wx/\Sigma w))^2 / \Sigma\, w-1$
Skewness	$\Sigma(x - (\Sigma x/n))^3$ / ($\Sigma(x - (\Sigma x/n))^2$)$^{3/2}$	$\Sigma\, w\,(x - (\Sigma wx/\Sigma w))^3$ / ($\Sigma\, w\,(x - (\Sigma wx/\Sigma w))^2$)$^{3/2}$
Kurtosis	$\Sigma(x - (\Sigma x/n))^4$ / ($\Sigma(x - (\Sigma x/n))^2$)^2 − 3	$\Sigma\, w\,(x - (\Sigma wx/\Sigma w))^4$ / $\Sigma\, w\,(x - (\Sigma wx/\Sigma w))^2$)^2 − 3
s.e. skewness	$\sqrt{}$({ $6n \times (n-1)$ } / { $(n-2) \times (n+1) \times (n+3)$ })	$\sqrt{}$({ $6\Sigma w \times (\Sigma w - 1)$ } / { $(\Sigma w - 2) \times (\Sigma w + 1) \times (\Sigma w + 3)$ }) (x not missing)
s.e. kurtosis	$\sqrt{}$({ $24 \times n \times (n-1)^2$ } / { $(n-2) \times (n-3) \times (n+5) \times (n+3)$ })	$\sqrt{}$({ $24 \times \Sigma w \times (\Sigma w - 1)^2$ } / { $(\Sigma w - 2) \times (\Sigma w - 3) \times (\Sigma w + 5) \times (\Sigma w + 3)$ }) (x not missing)

A quick look at the formula used for the weighted variance (or the standard deviation) or skewness or kurtosis shows that it breaks down for $\Sigma w<1$; in fact it is valid only when the weights are integer values greater than or equal to zero. Similarly, with quantiles the weights are assumed to specify replicated observations; so these must also be non-negative integers. If an invalid weight is found during the calculation of a variance, skewness, kurtosis or quantile a fault will be reported. Temporary tables will be deleted, but named tables may contain partial results. However, non-integer weights are allowed in other contexts. The standard deviation is the square root of the variance, and the standard error of the mean is the standard deviation divided by the square root of the number of observations.

If you have many observations to summarize, there may be insufficient space within Genstat for you to read them all and then form the tables. To cater for such situations, Genstat allows you to process the data in sections, using the SEQUENTIAL option of TABULATE in conjunction with the SEQUENTIAL option of READ. After READ, the absolute value of the option indicates the number of units that have been read in this particular section; the value is positive during interim sections and negative or zero once the terminator at the end of the data is reached. TABULATE will not print any tables until the final section has been processed. If you want to see the

intermediate tables, you can include a PRINT statement after the TABULATE statement. To allow Genstat to keep contact with the working tables in which the results are accumulating, you must save at least one out of the various types of table for every DATA variate. Genstat can then link the working tables to this named table during the course of the sequential tabulation, so that the information is not lost between the successive uses of TABULATE.

The final five options of TABULATE (OWN, OWNFACTORS, OWNVARIATES, INCHANNEL and INFILETYPE) allow you to link your own Fortran subroutine, G5XZIT, to Genstat to allow you to handle complicated arrangements of data, as can occur for example in hierarchical surveys. To implement this, you must get access to some of the Genstat source code. The relevant section of the code is named Module X, and is distributed with Genstat to all sites, probably in a file called X.FOR. The documentation of G5XZIT is included with the Fortran and so is not repeated here. G5XZIT is thus a Fortran subprogram, to be modified by you, which is called from within TABULATE for each unit to be tabulated. It contains switches to tell TABULATE when a data error occurs or when all the data have been read. To use it you have to link your own version of Genstat, as when using the OWN directive. Then your version of G5XZIT will be used instead of the standard version supplied as part of Genstat.

The subprogram can be as simple or as complicated as you like (or need), provided it obeys a few simple rules. A very simple version, reading two variates and two factors, is supplied with Genstat. This should provide sufficient information for you to write your own version, and link it into your own private version of Genstat.

The OWN option should be set to a variate allowing you to communicate between your Genstat code and your G5XZIT subprogram. The OWNFACTORS option provides the list of factors to be read by G5XZIT. It must include the classifying factors needed in the current TABULATE instruction, but it may contain others as well. The OWNVARIATES option should provide a similar list of variates. The INCHANNEL option should be set to the Genstat channel number of the data file, as specified in a previous OPEN statement or in the Genstat command line. The INFILETYPE option specifies whether the data file is character (input) or binary (unformatted).

TABULATE allows only one classification set to be used at a time. If the data set is complicated enough to require G5XZIT, then several tabulations with different classifying sets are likely to be needed. Rather than have a separate branch in G5XZIT for each tabulation, you can put all the factors and all the variates that you will need into the settings of the OWNFACTORS and OWNVARIATES options, and leave TABULATE to extract the ones it needs each time. If you have several TABULATE statements as suggested, you will have to close the data file and re-open it between them.

Options: PRINT, CLASSIFICATION, COUNTS, SEQUENTIAL, MARGINS, IPRINT, WEIGHTS, PERCENTQUANTILES, OWN, OWNFACTORS, OWNVARIATES, INCHANNEL, INFILETYPE.
Parameters: DATA, TOTALS, NOBSERVATIONS, MEANS, MINIMA, MAXIMA, VARIANCES, QUANTILES, SDS, SKEWNESS, KURTOSIS, SEMEANS, SESKEWNESS, SEKURTOSIS.

Action with RESTRICT

If any of the DATA variates, or the WEIGHTS variate, or any of the classifying factors of the tables is restricted, TABULATE will form the tables using only the defined subset of units. If more than one variate or factor is restricted, the restrictions must be the same.

See also

Directives: TABLE, MARGIN, COMBINE.

Procedures: FBETWEENGROUPVECTORS, MTABULATE, PERCENT, SVSTRATIFIED, SVTABULATE, TABINSERT, TABMODE, TABSORT, VSUMMARY.

Genstat Reference Manual 1 Summary sections on: Basic and nonparametric statistics, Calculations and manipulation, Survey analysis.

TDISPLAY

Displays further output after an analysis by TFIT.

Options

PRINT = *string tokens*	What to print (model, summary, estimates, correlations); default mode,summ,esti
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
SAVE = *identifier*	Save structure to supply fitted model; default * i.e. that from the last model fitted

No parameters

Description

You can use TDISPLAY to print further output from an TFIT statement. The PRINT option has the same interpretation as in TFIT, except that information is not available to monitor convergence. Also, if the TFIT statement used the setting METHOD=initialize you will not be able to print the standard errors or correlations between the parameter estimates.

The CHANNEL option allows you to send the output to another output channel.

You can use the SAVE option to specify the time-series save structure (from TFIT) from which the output is to be taken. By default TDISPLAY uses the structure from the most recent TFIT statement.

Options: PRINT, CHANNEL, SAVE.
Parameters: none.

See also

Directives: TSM, FTSM, TFILTER, TFIT, TFORECAST, TKEEP, TRANSFERFUNCTION, TSUMMARIZE.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

TERMS

Specifies a maximal model, containing all terms to be used in subsequent linear, generalized linear, generalized additive and nonlinear models.

Options

PRINT = *string tokens*	What to print (correlations, wmeans, SSPM, monitoring); default *
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (yes, no); default no
SSPM = *SSPM*	Gives sums of squares and products on which to base calculations; default *
TOLERANCE = *scalar*	Criterion for testing for linear dependence; default is $10^7\varepsilon$, where ε is the smallest real value such that $1+\varepsilon$ is greater than 1 on the computer
DESIGNMATRIX = *matrix*	Saves the design matrix for the maximal model
MVINCLUDE = *string token*	Whether to include units with missing values in the explanatory factors and variates (explanatory); default * i.e. omit these
RIDGE = *scalar* or *variate*	Supplies values to add to the diagonal of the sums-of-squares-and-products matrix, to enable ridge methods to be used; default 0
CLDESIGNMATRIX = *text*	Saves the column labels of the design matrix for the maximal model i.e. the names of the parameters estimated in the maximal model
CLSSP = *text*	Saves the labels of the sum-of-squares-and-products matrix

Parameter

formula	List of explanatory variates and factors, or model formula

Description

You can use the TERMS directive before starting to explore different subsets of explanatory variables, to allow Genstat to define a common set of units for the regression and to carry out some initial calculations. TERMS thus initializes Genstat ready for an exploration using the directives ADD, DROP, SWITCH, TRY or STEP. It overrules any model that has already been fitted with FIT, FITCURVE or FITNONLINEAR and resets the current model to be the null model containing only the constant term.

TERMS need not be specified before exploring a linear, generalized linear or generalized additive model, that is one that is fitted initially using FIT with its CALCULATION option unset. However, it is essential before exploring a nonlinear model, that is one fitted initially by FIT with CALCULATION set, or by FITCURVE or FITNONLINEAR. Furthermore, if some of the explanatory variables to be used in a linear, generalized linear or generalized aditive model contain missing values or have restrictions, the use of TERMS ensures that the sequence of models are fitted using a common set of units. Otherwise, if a variate or factor which is introduced into the model has a missing value where previous explanatory variates or factors did not, or is restricted whereas previous ones were not, the set of units has to be changed. The previous model is automatically refitted with the new set of units before the new model is fitted, but the accumulated summary will then show only these two fits.

The formula specified by the parameter of TERMS should contain all the explanatory variables

and model terms that you may wish to use in the subsets. The model containing all the terms specified in the formula, excluding the response variates, is called the *maximal model*.

The calculations are weighted if you have specified weights in the MODEL statement, and they are made within groups if you have specified a grouping factor. All units of the variates are used unless there are restrictions or missing values. Genstat will look for restrictions on response variates, explanatory variates, the weight variate, the offset variate and the grouping factor (but these must not be restricted in different ways). A missing value in any of these structures except a response variate will also exclude the corresponding unit.

The PRINT option controls printed output, with settings:

SSPM	sums of squares and products between the variates in the model (including the response variates and dummy variates set up to represent any factors and their interactions), the means of the variates and the degrees of freedom;
correlation	the matrix of correlations between variables;
wmeans	group means for a within-group regression;
monitoring	monitoring information from the fit of the null model.

The FACTORIAL option controls the inclusion of interaction terms in the model. All terms involving more than the specified number of factors and variates are omitted. By default FACTORIAL is set to three. The FULL option can be set to yes to ensure that Genstat allocates a parameter to every level of each factor in a linear, generalized linear or generalized additive model; otherwise it will exclude the reference level of the factor (and its estimates will represent the differences between the estimated parameters for the other levels and the estimate for the reference level).

The SSPM option lets you use values that you have already calculated for an SSPM structure. This is feasible only with ordinary linear regression but it can be useful when you are analysing very large sets of data: you can accumulate an SSPM sequentially with the FSSPM directive to avoid storing all the data at one time. Later regression calculations will be based on the supplied values of the SSPM, though no fitted values, residuals or leverages will be available. However, the values of a supplied SSPM are accepted without checking by the TERMS directive: Genstat simply assumes you are giving it something sensible.

The TOLERANCE option controls the detection of aliasing in subsequent model fitting. By default, a parameter in a linear or generalized linear model will be deemed to be aliased if the ratio between the original diagonal value of the SSPM corresponding to this parameter and the current diagonal value of the partially inverted SSPM is less than $10^7\varepsilon$. The quantity ε depends on the computer and is defined to be the smallest number such that the computer recognizes $1.0 + \varepsilon$ as greater than 1.0 in double precision. Any positive value can be supplied by the TOLERANCE option to replace this default criterion in subsequent linear regression and generalized linear regression.

The DESIGNMATRIX option can be set to a matrix to save the design matrix corresponding to the maximal model. The CLDESIGNMATRIX option can save the column labels of the design matrix without saving the design matrix itself. (These are the names of the parameters estimated in the maximal model.)

The MVINCLUDE option allows units with missing values with missing values in factors or variates in the model to be included (by default these are excluded). Where this occurs, the factor or variate is taken to make no contribution to the fitted value for the unit concerned. This is an option that should be set only under very special circumstances, for example it is required internally by some of the procedures that fit hierarchical generalized linear models (see HGANALYSE). It should **not** be used during ordinary analyses.

The RIDGE option enables ridge methods to be implemented. It can be set to a scalar, to define a constant to add to all the diagonal elements of the sums-of-squares-and-products matrix that correspond to the parameters in the model. Alternatively you can set RIDGE to a variate, to add

a different value to each diagonal element. You may then want to use the CLSSP option to save the row labels of the sum-of-squares-and-products matrix, so that you see which rows correspond to model parameters, and which ones correspond to the y-variates. By default nothing is added (i.e. RIDGE = 0).

Options: PRINT, FACTORIAL, FULL, SSPM, TOLERANCE, DESIGNMATRIX, MVINCLUDE, RIDGE, CLDESIGNMATRIX, CLSSP.
Parameter: unnamed.

Action with RESTRICT
You can restrict the units that Genstat will use for the regression by putting a restriction on any of the vectors involved in the MODEL statement (response variates, weight variate, offset variate, grouping factor or variate of binomial totals), or on any explanatory variate or factor in the TERMS statement. However, you are not allowed to have different restrictions on the different vectors. You should not alter the restriction applied to the vectors between the TERMS statement and subsequent fitting statements.

See also
Directives: FIT, FITCURVE, FITNONLINEAR, MODEL.
Genstat Reference Manual 1 Summary section on: Regression analysis.

TEXT

Declares one or more text data structures.

Options

NVALUES = *scalar* or *vector*	Number of strings, or vector of labels; default * takes the setting from the preceding UNITS statement, if any
VALUES = *strings*	Values for all the texts; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the texts in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the texts
VALUES = *texts*	Values for each text
CHARACTERS = *scalars*	Numbers of characters of the lines of each text to be printed by default
EXTRA = *texts*	Extra text associated with each identifier

Description

Each unit of a Genstat text structure is a string which you can regard as a line of textual description. Texts can be used to label vectors and pointers, for captions or pieces of explanation within output, to store Genstat statements and to store output.

The IDENTIFIER parameter lists the texts that are to be declared. Values can be assigned to the texts by either the VALUES option or the VALUES parameter. The option defines a common value for all the texts in the declaration, while the parameter allows them each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

The NVALUES option allows the number of values in the texts to be defined. If this is not set, the lengths of the texts are defined from the numbers that are supplied by the VALUES option or parameter. If these too are unset, Genstat takes the length specified by the preceding UNITS statement, if any.

The CHARACTERS parameter allows you to define the number of characters to be printed by default when the strings of each text are printed. You can associate a text of extra annotation with each table using the EXTRA parameter.

If the MODIFY option is set to yes any existing attributes and values of the texts are retained (if still appropriate); otherwise these are lost. The IPRINT option can be set to specify how the texts will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself).

The text can contain any of the characters that you can generate on your computer. The text has an internal logical attribute, known as *coding*, to indicate whether it contains characters like Chinese, Korean or Thai characters, which need to be coded internally in a more complicated way than ordinary letters, numbers etc. (Technically they are stored as multi-byte UTF-8 characters.) Some applications may not be able to display output containing these characters successfully. You can access the coding attribute using the GETATTRIBUTE directive.

Options: NVALUES, VALUES, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUES, CHARACTERS, EXTRA.

See also

Directives: CONCATENATE, EDIT, TXBREAK, TXCONSTRUCT, TXFIND, TXINTEGERCODES, TXPOSITION, TXREPLACE, TX2VARIATE, FACTOR, VARIATE, UNITS.

Procedures: TXPAD, TXPROGRESSION, TXSPLIT.

Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.

Genstat Reference Manual 1 Summary section on: Data structures.

TFILTER

Filters time series by time-series models.

Option

PRINT = *string tokens*	What to print (series); default *

Parameters

OLDSERIES = *variates*	Time series to be filtered
NEWSERIES = *variates*	To save filtered series
FILTER = *TSMs*	Models to filter with respect to
ARIMA = *TSMs*	ARIMA models for time series

Description

Filtering is a means of processing a time series in order to produce a new series. The purpose is usually to reveal some features and remove other features of the original series. Filters in Genstat are one-sided: that is, each value in the new series depends only on present and past values of the original series. However, you can do two-sided filtering by using the SHIFT and REVERSE functions of CALCULATE.

TFILTER was originally called FILTER, but was renamed in Release 14 to emphasize its status as a time-series command. The earlier name (FILTER) was retained to allow previous programs to continue to run, but this may be removed in a future release.

The OLDSERIES and NEWSERIES parameters of TFILTER specify respectively the time series to be filtered, and the series that result from filtering. A new series must not have the same identifier as the series from which it was calculated. Genstat interprets any missing values in the old series as zero. But if you use the ARIMA parameter (see below), Genstat replaces them by interpolated values when it calculates the filtered series; the missing values remain in the old series.

The FILTER parameter specifies the TSMs to be used for filtering. If the TSM is a transfer-function model, the new series y_t is calculated from the old series x_t by

$$y_t = \{ \omega(B)B^b / \delta(B)\nabla^d \}\ x_t.$$

The filter does not use the power transformation nor the reference constant. This lets you apply a single filter conveniently to a set of time series, for which different transformations and different constants might be appropriate. You can always use the CALCULATE directive to apply a transformation to a series before using TFILTER.

If the TSM is an ARIMA model, then the new series a_t is calculated from the old series y_t by

$$a_t = \{ \varphi(B)\nabla^d / \theta(B)\}\ y_t.$$

Note that the TSM does not have to be the model appropriate for y_t. Again, Genstat ignores the parameters λ, c and σ_a^2; you can set them to 1,0,0, for example.

The ARIMA parameter specifies a time-series model for the old series. The purpose is to reduce transient errors that arise in the early part of the new series: these arise because Genstat does not know the values of the old series that came before those that have been supplied. If you do not use this parameter, then Genstat takes these earlier values to be zero. This can cause unacceptable transients which can only be partially removed by procedures such as mean-correcting the old series. If you do use the ARIMA parameter, then Genstat uses the specified model to estimate (or back-forecast) the values of the old series earlier than those that have been supplied.

You do not have to have a good ARIMA model for the old series in order to achieve worthwhile reductions in the transients. Thus a model with orders (0,1,1) and parameters (1,0,0,0.7) would estimate the prior values to be constant, at a level that is a backward EWMA of the early values of the series.

For a seasonal monthly time series, an appropriate ARIMA model could have orders

(0,1,1,0,1,1,12) and parameters (1,0,0,0.7,0.7). However you must give the supplied model a transformation parameter λ=1. Any other value for λ breaks the assumption of linearity that underlies the calculations for correcting the transients. The constant term in the ARIMA model can be non-zero, and should be if that is appropriate for the old series. Note that the ARIMA model does not define the filter.

If you specify the ARIMA parameter, Genstat uses this model to interpolate any missing values in the old series before it calculates the new series. Suppose for example that the filter is the identity, defined by a transfer-function model with orders (0,0,0,0) and parameters (1,0,0); then the new series will be the old series with any missing values replaced.

Option: PRINT.
Parameters: OLDSERIES, NEWSERIES, FILTER, ARIMA.

Action with RESTRICT
The OLDSERIES variate can be restricted, but this must be to a contiguous set of units.

See also
Directives: TSM, FTSM, TDISPLAY, TFIT, TFORECAST, TKEEP, TRANSFERFUNCTION, TSUMMARIZE, CORRELATE, FOURIER.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

TFIT

Estimates parameters in Box-Jenkins models for time series.

Options

PRINT = *string tokens*	What to print (model, summary, estimates, correlations, monitoring); default mode,summ,esti
LIKELIHOOD = *string token*	Method of likelihood calculation (exact, leastsquares, marginal); default exac
CONSTANT = *string token*	How to treat the constant (estimate, fix); default esti
RECYCLE = *string token*	Whether to continue from previous estimation (yes, no); default no
WEIGHTS = *variate*	Weights; default *
MVREPLACE = *string token*	Whether to replace missing values by their estimates (yes, no); default no
FIX = *variate*	Defines constraints on parameters (ordered as in each model, tf models first): zeros fix parameters, parameters with equal numbers are constrained to be equal; default *
METHOD = *string token*	Whether to carry out full iterative estimation, to carry out just one iterative step, to perform no steps but still give parameter standard deviations, or only to initialize for forecasting by regenerating residuals (full, onestep, zerostep, initialize); default full
MAXCYCLE = *scalar*	Maximum number of iterations; default 15
TOLERANCE = *scalar*	Criterion for convergence; default 0.0004
SAVE = *identifier*	To name save structure, or supply save structure with transfer-functions; default * i.e. transfer-functions taken from the latest model

Parameters

SERIES = *variate*	Time series to be modelled (output series)
TSM = *TSM*	Model for output series
BOXCOXMETHOD = *string token*	How to treat transformation parameter in output series (fix, estimate); default fix
RESIDUALS = *variate*	To save residual series

Description

The main use of TFIT is to fit parameters to time-series models, although you can also use it to initialize for the TFORECAST directive, even when the model parameters are already known. TFIT was originally called ESTIMATE, but was renamed in Release 14 to emphasize its status as a time-series command. The earlier name (ESTIMATE) was retained to allow previous programs to continue to run, but this may be removed in a future release.

You need to define a TSM structure before using TFIT, to provide the setting for the TSM parameter. You may also wish to give a TRANSFERFUNCTION statement, for example if you wish to specify explanatory variables for regression with ARIMA errors, or to define transfer-function models. In many applications of estimating a univariate ARIMA model, you will need only a simple form of the directive, such as:

```
TFIT Daylength; TSM=Erp
```

The SERIES parameter specifies the variate holding the time series data to which the model

is to be fitted.

The TSM parameter specifies the ARIMA model that is to be fitted to the time-series data. This TSM must already have been declared and its ORDERS must have been set. If the LAGS parameter of the TSM has been set, the lags must have been given values. However, if the PARAMETERS of the TSM model have been set, these need not have been declared previously nor given values. When the parameter values are not set, default values are used: these are all zero, except for the transformation parameter, which is set to 1.0 if it is not to be estimated (see BOXCOXMETHOD and FIX below). Any parameter values that you do specify will be used as initial values for the parameters in the model; Genstat replaces any missing values by the default values. If any group of autoregressive or moving-average parameters do not satisfy the required conditions for stationarity or invertibility, all the parameters to be estimated are reset by Genstat to the default values. After TFIT, the parameters of the TSM contain the estimated parameter values.

The BOXCOXMETHOD parameter allows you to estimate the transformation parameter λ.

The RESIDUALS parameter saves the estimated innovations (or residuals). The residuals are calculated for $t=t_0...N$, where $t_0=1+p+d-q$ for a simple ARIMA model. If $t_0>1$, missing values will be inserted for $t=1...t_0-1$.

The PRINT option controls printed output. If you specify monitoring, then at each cycle of the iterative process of estimation, Genstat prints the *deviance* for the current fitted model, together with the current estimates of model parameters. The format is simple with the minimum of description, to let you judge easily how quickly the process is converging. The other settings of PRINT control output at the end of the iterative process. If you specify model, the model is briefly described, giving the identifier of the series and the time-series model, together with the orders of the model. If you specify summary, the deviance of the final model is printed, along with the residual number of degrees of freedom. If you specify estimates, the estimates of the model parameter are printed in a descriptive format, together with their estimated standard errors and reference numbers. If you specify correlations, the correlations between estimates of parameters are printed, with reference numbers to identify the parameters.

The LIKELIHOOD option specifies the criterion that Genstat minimizes to obtain the estimates of the parameters: this is described in the next section. The default setting exact is recommended for most applications.

You can use the CONSTANT option to specify whether Genstat is to estimate the constant term c in the model. If CONSTANT=fix, the constant is held at the value given in the initial parameter values; this need not be zero.

The RECYCLE option allows a previous TFIT statement to continue; this can save computing time. If RECYCLE=yes, the most recent TFIT statement is continued, unless the SAVE option has been set to the save structure from some other TFIT statement. The SERIES and TSM settings are then taken from this previous TFIT statement: Genstat ignores any specified in the current statement. Most of the settings of other parameters and options are carried over from the previous statement, and new values are ignored. However, there are some exceptions. You can change the RESIDUALS variate, you can reset MAXCYCLE to the number of further iterations you require, and you can change the settings of TOLERANCE and PRINT. You can also change the values of the variate in the WEIGHTS option; you can thus get reweighted estimation. You can change the values of the SERIES itself, although you cannot change missing values; if the MVREPLACE option was previously set to yes, you must put the original missing values back into the SERIES variate before the new TFIT statement.

The WEIGHTS option includes in the likelihood a weighted sum-of-squares term

$$\sum_{t=t0...N} \{ w_t a_t^2 \}$$

where w_t, $t=1...N$ are provided by the WEIGHTS variate. The values of w_t must be strictly positive. If $t_0<1$, where $t_0=1+d+p-q$, then w_t is taken as 1 for $t<1$.

The MVREPLACE option allows you to request any missing values in the time-series to be replaced by their estimates after estimation. Genstat will always estimate the missing values,

irrespective of the setting of MVREPLACE; so you can also obtain these estimates later from TKEEP.

The FIX option allows you to place simple constraints on parameter values throughout the estimation. The units of the FIX variate correspond to the parameters of the TSM, excluding the innovation variance. The values of the FIX variate are used to define the parameter constraints and must be integers. If an element of the FIX variate is set to 0, the corresponding parameter is constrained to remain at its initial setting. If an element is not 0, and the value is unique in the FIX variate, the parameter is estimated without any special constraint. If two or more values are equal, the corresponding parameters are constrained to be equal throughout the estimation. The number that you give to a parameter by FIX will appear as the reference number of the parameter in the printed model and correlation matrix. This option overrides any setting of CONSTANT and BOXCOXMETHOD.

The MAXCYCLE option specifies the maximum number of iterations to be performed.

The TOLERANCE option specifies the convergence criterion. Genstat decides that convergence has occurred if the fractional reduction in the deviance in successive iterations is less than the specified value, provided also that the search is not encountering numerical difficulties that force the step length in the parameter space to be severely limited. You can use monitoring to judge whether, for all practical purposes, the iterations have converged. Genstat gives warnings if the specified number of iterations is completed without convergence, or if the search procedure fails to find a reduced value of the deviance despite a very short step length. Such an outcome may be due to complexities in the likelihood function that make the search difficult, but can be due to your specifying too small a value for TOLERANCE.

The SAVE option allows you to save the *time-series save structure* produced by TFIT. You can use this in further TFIT statements with RECYCLE=yes, or in TFORECAST statements. It can also be used by the TDISPLAY and TKEEP directives. Genstat automatically saves the structure from the most recent TFIT statement, but this is over-written when the next TFIT statement is executed, unless you have used SAVE to give it an identifier of its own. You can access the current time-series save structure by the SPECIAL option of the GET directive, and reset it by the TSAVE option of the SET directive.

The METHOD option has four possible settings. The default setting is full which gives the usual estimation to convergence or until the maximum number of iterations has been reached.

With the setting METHOD=initialize, TFIT carries out only the residual regeneration steps (that is, calculation of a_t for $t=t_0...N$) which are needed before TFORECAST can be used. If the model has just been estimated using the default full setting, this is unnecessary. The setting initialize is useful when the time series is supplied with a known model and a minimal amount of calculation is wanted to prepare or initialize for forecasting. None of the model parameters are changed, and no standard errors of parameter estimates are available. Missing values in the series *are* estimated so this setting provides an efficient way of getting their values when the time series model is known; they can then be obtained using TKEEP. The deviance value is also available from TKEEP. This setting is therefore useful for efficient calculation of deviance values when you want to plot the shape of the deviance as a function of parameter values.

With the setting METHOD=zerostep the effect is the same as for initialize except that TFIT also calculates the standard errors of the parameters as if they had just been estimated. These can be used together with other quantities available from TKEEP to construct confidence intervals and carry out tests on the parameter values, which remain unchanged except that the innovation variance in the ARIMA model is replaced by its estimate conditional on all other parameters.

The setting METHOD=onestep gives the same results as specifying the option MAXCYCLE=1 in TFIT. It is convenient for carrying out quick tests of model parameters.

To explain the LIKELIHOOD option, we need to describe the estimation of ARIMA models

in more detail. You may want to skip this if you are doing fairly routine work.

The first step in deriving the likelihood for a simple model is to calculate

$$w_t = \nabla^d y_t - c\,, \qquad t = 1+d \ldots N$$

This has a multivariate Normal distribution with dispersion matrix $V\sigma_a^2$, where V depends only on the autoregressive and moving-average parameters. The likelihood is then proportional to

$$\{ \sigma_a^{2m} \left| V \right| \}^{-½} \exp\{ -w'V^{-1}w/2\sigma_a^2 \}$$

where $m=N-d$. In practice Genstat evaluates this by using the formula

$$w'\, V^{-1}\, w \;=\; W + \sum_{t=t0 \ldots N} \{ a_t^2 \} \;=\; S$$

where $t_0=1+d+p-q$. The term W is a quadratic form in the p values $w_{1+d-q} \ldots w_{p+d-q}$: it takes account of the starting-value problem for regenerating the innovations a_t, and avoids losing information as would happen if the process used only a conditional sum-of-squares function. If $q>0$, Genstat introduces unobserved values of $w_{1+d-q} \ldots w_d$ in order to calculate the sum S. Genstat uses linear least-squares to calculate these q starting values for w, thus minimizing S. We shall call them *back-forecasts*, though if $p>0$ they are actually computationally convenient linear functions of the proper back-forecasts. We shall call S the sum-of-squares function: it is the sum of the quadratic form and the sum-of-squares term, and is identical to the value expressed by Box & Jenkins (1970) as

$$\sum_{t=-\infty \ldots N} \{ a_t^2 \}$$

using infinite back-forecasting; that is, using:

$$W \;=\; \sum_{t=-\infty \ldots t0-1} \{ a_t^2 \}$$

The values a_t for $t=t_0 \ldots N$ agree precisely with those of Box and Jenkins.

To clarify all this, consider examples with no differencing; that is, $d=0$. If $p=0$ and $q=1$ then $W=0$ and $t_0=0$, and one back-forecast w_0 is introduced. If $p=1$ and $q=0$ then $W=(1-\varphi_1^2)w_1^2$ and $t_0=2$, and no back-forecasts are needed. If $p=q=1$ then $W=(1-\varphi_1^2)w_0^2$ and $t_0=1$, and so one back-forecast w_0 is needed. In this case the proper back-forecast is in fact $w_0/(1-\theta_1\varphi_1)$.

The value of $\left| V \right|$ is a by-product of calculating W and the back-forecast. For example, if $p=0$ and $q=1$, then

$$\left| V \right| = (1 + \theta_1^2 + \ldots + \theta_1^{2N})$$

If $p=1$ and $q=0$,

$$\left| V \right| = 1 / (1 - \varphi_1^2)$$

and if $p=q=1$,

$$\left| V \right| = 1 + (\varphi_1 - \theta_1)^2 (1 + \theta_1^2 + \ldots + \theta_1^{2N-2}) / (1 - \varphi_1^2)$$

Concentrating the likelihood over σ_a^2 by setting $\sigma_a^2=S/m$ yields a value proportional to $\{ \left| V \right|^{1/m} S \}^{-m/2}$.

The default setting of the `LIKELIHOOD` option is `exact`. In this case the concentrated likelihood is maximized, by minimizing the quantity

$$D = \left| V \right|^{1/m} S$$

which is called the deviance.

The setting `leastsquares` specifies that Genstat is to minimize only the sum-of-squares term S. This criterion corresponds to the back-forecasting sum-of-squares used by Box & Jenkins (1970), and will in many cases give estimates close to those of the exact likelihood. However, some discrepancy arises if the series is short or the model is close to the invertibility boundary. This is because of limitations on the back-forecasting procedure, as described in the algorithms of Box & Jenkins (1970). The deviance value D that Genstat prints is, with this setting, simply S.

When you use exact likelihood, the factor $\left| V \right|^{1/m}$ reduces bias in the estimates of the parameter; you would get bias if you used `leastsquares` instead. However, $\left| V \right|^{1/m}$ is generally close to one, unless the series is short or the model is either seasonal or close to the boundaries of invertibility or stationarity. The `leastsquares` setting is therefore adequate for most long, non-seasonal sets of data; using it may reduce the computation time by up to 50%. When you specify that Genstat is to estimate the parameter λ of the Box-Cox transformation, Genstat also

includes the Jacobian of the transformation in the likelihood function. The result is an extra factor $G^{-2(\lambda-1)}$ in the definition of the deviance, G being the geometric mean of the data,

$$G = (\prod_{t=1 \ldots N} \{ y_t \}) ** (1 / N)$$

Note that this is not included unless λ is being estimated, even if $\lambda \neq 1$.

You can treat differences in $N\log(D)$ as a chi-square variable in order to test nested models: this is supported by asymptotic theory, and by experience with models that have moderately large sample sizes. Similarly, you can select between different models by using $N\log(D)+2k$ as an information criterion, k being the number of estimated parameters. But both of these test procedures are questionable if the estimated models are close to the boundaries of invertibility or stationarity. Provided all the models that are being compared have the same orders of differencing, with the differenced series being of length m, it is recommended that $m\log(D)$ be used rather than $N\log(D)$ in these tests since $m\log(D)$ is precisely minus two multiplied by the log-likelihood as defined above.

The setting `marginal` is relevant mainly when `TFIT` is used for regression with ARIMA errors. (This requires a `TRANSFERFUNCTION` statement beforehand to specify the explanatory variables.) The likelihood for the model is defined as that of the univariate error series e_t which is defined in general by

$$e_t = y_t - b_1 x_{1,t} - \ldots - b_m x_{m,t}$$

(the x_i being m explanatory variables). The constant term therefore appears in the model after any differencing of e_t; for example

$$\nabla e_t = c + (1 - \theta_1 B) a_t$$

You can get bias in the estimates of the parameters of an ARIMA model because the regression is estimated at the same time. You can guard against this by specifying `LIKELIHOOD=marginal`. This can be particularly important if the series are short or if you use many explanatory variables (Tunnicliffe Wilson 1989). The deviance is now defined as

$$D = S (\left| X' V^{-1} X \right| \left| V \right|)^{1/m}$$

where m is reduced by the number of regressors (including the constant term) and the columns of X are the differenced explanatory series: the other terms are as in the exact likelihood.

You can use the `marginal` setting also for univariate ARIMA modelling, when the constant term is the only explanatory term. Furthermore, Genstat deals with missing values in the response variate by doing a regression on indicator variates; these too are included in the X matrix. However, you cannot use marginal likelihood and estimate a transformation parameter in either the transfer-function model or an ARIMA model. Neither can you use it if you set the `FIX` option in `TFIT`. In these cases Genstat automatically resets the `LIKELIHOOD` option to `exact`.

At every iteration with the setting `LIKELIHOOD=marginal`, the regression coefficients are the maximum-likelihood estimates conditional upon the estimated values of the parameters of the ARIMA model: these are also the generalized least-squares estimates, conditioned in the same way. This is so even if `MAXCYCLE=0`; that is, the coefficients of the regression are re-estimated even at iteration 0. Therefore you must not use the `marginal` setting with the option `METHOD=initialize` to initialize for `TFORECAST`. You can compare deviance values that were obtained using marginal likelihood only for models with the same explanatory variables and the same differencing structure in the error model.

Options: `PRINT`, `LIKELIHOOD`, `CONSTANT`, `RECYCLE`, `WEIGHTS`, `MVREPLACE`, `FIX` , `METHOD`, `MAXCYCLE`, `TOLERANCE`, `SAVE`.
Parameters: `SERIES`, `TSM`, `BOXCOXMETHOD`, `RESIDUALS`.

Action with `RESTRICT`
The `SERIES` variate can be restricted, but this must be to a contiguous set of units.

References

Box, G.E.P. & Jenkins, G.M. (1970). *Time Series Analysis, Forecasting and Control*. Holden-Day, San Francisco.

Tunnicliffe Wilson, G. (1989). On the use of marginal likelihood in time-series model estimation. *Journal of the Royal Statistical Society, Series B*, **51**, 15-27.

See also

Directives: TSM, FTSM, TRANSFERFUNCTION, TDISPLAY, TFILTER, TFORECAST, TKEEP, TSUMMARIZE, CORRELATE, FOURIER.

Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY, MOVINGAVERAGE, PERIODTEST, PREWHITEN, REPPERIODOGRAM, SMOOTHSPECTRUM.

Genstat Reference Manual 1 Summary section on: Time series.

TFORECAST

Forecasts future values of a time series.

Options

PRINT = *string tokens*	What to print (forecasts, limits, setransform, sfe); default fore,limi
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
ORIGIN = *scalar*	Number of known values to be incorporated; default 0
UPDATE = *string token*	Whether to update the forecast origin to the end of the new observations (yes, no); default no
NEWOBSERVATIONS = *variate*	Variate of length ≥ ORIGIN providing new values of the time series to be incorporated (must be set if ORIGIN > 0)
SFE = *variate*	Saves standardized forecast errors; default *
MAXLEAD = *scalar*	Maximum lead time i.e number of forecasts to be made; default * defines the number as the length of FORECAST variate
FORECAST = *variate*	Variate of length MAXLEAD to save forecasts of output series; default *
SETRANSFORM = *variate*	Saves standard errors of the forecasts (on transformed scale, if defined); default *
LOWER = *variate*	Saves lower confidence limits; default *
UPPER = *variate*	Saves upper confidence limits; default *
PROBABILITY = *scalar*	Probability level for confidence limits; default 0.9
COMPONENTS = *pointer*	Contains variates (of length ORIGIN + MAXLEAD) to save components of the forecast
SAVE = *identifier*	Save structure to supply fitted model; default * i.e. that from last model fitted

Parameters

FUTURE = *variates*	Variates (of length ORIGIN + MAXLEAD) containing future values of input series
METHOD = *string tokens*	How to treat future values of input series (observations, forecasts); default obse

Description

TFORECAST can be used after TFIT to forecast future values of a time series. TFORECAST was originally called FORECAST, but was renamed in Release 14 to emphasize its status as a time-series command. The earlier name (FORECAST) was retained to allow previous programs to continue to run, but this may be removed in a future release.

In many applications of forecasting with univariate ARIMA models, you will need only a simple form of the directive. For example

```
TFORECAST [MAXLEAD=10]
```

will cause Genstat to print forecasts for 10 lead times, that is, the next 10 time points after the end of your data. However, you must already have used TFIT to specify the time series to be forecast, and the model to be used for forecasting. This information is supplied by the SAVE option; if SAVE is not specified, TFORECAST uses the information from the most recent TFIT statement. Once you have used TFIT, you can give successive TFORECAST statements to incorporate new observations of the time series, and to produce forecasts from the end of the new data.

If the time series is supplied with a known model (that is, one with all its orders and parameters specified) you can use TFIT with option setting METHOD=initialize before you use TFORECAST. This will carry out just sufficient calculations, in particular the regeneration of the model residuals, for TFORECAST to be used. The model parameters will not be changed – not even the innovation variance. This setting of METHOD restricts the structures, such as parameter standard errors, that can be accessed using TDISPLAY and TKEEP after TFIT. The SAVE structure created by using TFIT with METHOD=initialize thus requires less space than that produced by the other settings.

The PRINT option controls the printed output, and the CHANNEL option allows this to be sent to another output channel.

The ORIGIN option specifies the number of new values to be incorporated before forecasting ahead from that point. Setting this to a positive value *n* indicates that *n* new observations are to be added onto the end of the series. These new observations must be supplied in a variate using the NEWOBSERVATIONS option, which must be of length $\geq n$. The standardized forecast errors for these new observations can be printed or saved in a variate of length *n* using the SFE option.

The UPDATE option specifies whether these new observations are to be incorporated internally onto the end of the time series and the internal pointer moved to the end of the new observations. If UPDATE=yes is used, then ORIGIN=0 in future calls to TFORECAST will point to the end of the *n* new observations. If the default, UPDATE=no is used, then the internal pointer remains at the end of the original series.

The number of future values to be forecast is set by option MAXLEAD. These new values can be saved in a variate of length MAXLEAD using the FORECAST option.

The PROBABILITY option determines the width of the error limits on the forecast. It defines the probability that the actual value will be contained within the limits at any particular lead time. Note that the limits do not apply simultaneously over all lead times.

The SETRANSFORM option specifies a variate to store the standard errors that Genstat used in calculating the error limits of the forecasts, starting at lead time 1. These are the standard errors of the transformed series, according to the value of the Box-Cox transformation parameter; they are functions of the model only, not of the data.

The LOWER option specifies a variate to store the lower limits of the forecasts. This must be the same length as the FORECAST variate. The TFORECAST directive puts the values of the lower limit into the variate, matching the forecasts in the FORECAST variate. The UPPER option similarly allows the upper limits to be saved. Note that the limits are constructed as symmetric percentiles, assuming Normality of the transformed time series. Similarly, the forecast is a median value – not necessarily the mode or the mean, unless the transformation parameter is 1.0.

The SFE option specifies a variate to save the standardized errors of the forecasts. These are the innovation values that are generated as each successive new observation is incorporated, divided by the square root of the TSM innovation variance. They provide a useful check on the continuing adequacy of the model. For example, excessively large values (compared to the standard Normal distribution) may indicate that you should revise the model. The variate must be the same length as the FORECAST variate. The TFORECAST directive places values of the errors in the variate, matching the new observations in the FORECAST variate.

The parameters of TFORECAST are relevant only when the time-series model incorporates explanatory variables, which requires a TRANSFERFUNCTION statement before the TFIT statement. You use the FUTURE parameter to specify a list of variates, corresponding to the list of variates specified by the SERIES parameter of TRANSFERFUNCTION. These variates must all have the same length. They hold future values of the explanatory variables to be used either for constructing forecasts of the response variable, or for incorporating new observations in order to revise the forecasts. You can use the METHOD parameter when some or all of the future values of the explanatory variables are forecasts obtained using univariate ARIMA models. You can amend the error limits of the forecasts for the response variable to allow for the uncertainty in

these future values, but you need to assume that there is no cross-correlation between the errors in these predictions. The list of strings specified by the METHOD parameter indicates for each explanatory variable whether such an allowance should be made. The future values of a series are by default treated as known values if no corresponding ARIMA model is present, or if the transformation parameter of the ARIMA model is not equal to the value used in the regression model for that series. You can change the settings of the METHOD parameter in successive TFORECAST statements.

The COMPONENTS option is also relevant only when the time-series model incorporates explanatory variables, and can be used to specify a pointer to variates in which you can save components of future values of the output series. There is a variate for each input component and for the output noise component. These variates correspond exactly to the variates that were specified by the FUTURE parameter for the input series, and by the FORECAST variate for the output series; corresponding lengths must match. The values that the variates hold can therefore be components of the forecasts of the output series, or can be new observations. The can be used to investigate the structure of forecasts.

If the input series ARIMA model and the transfer-function model have differing transformation parameters, then the METHOD option reverts to its default action of treating the values of any future input series as known quantities rather than forecasts.

Options: PRINT, CHANNEL, ORIGIN, UPDATE, NEWOBSERVATIONS, SFE, MAXLEAD, FORECAST, SETRANSFORM, LOWER, UPPER, PROBABILITY, COMPONENTS, SAVE.
Parameters: FUTURE, METHOD.

See also

Directives: TSM, FTSM, TRANSFERFUNCTION, TDISPLAY, TFILTER, TFIT, TKEEP, TSUMMARIZE.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

TKEEP

Saves results after an analysis by TFIT.

Option

SAVE = *identifier*	Save structure to supply fitted model; default * i.e. that from last model fitted

Parameters

OUTPUTSERIES = *variate*	Output series to which model was fitted
RESIDUALS = *variate*	Residual series
ESTIMATES = *variate*	Estimates of parameters
SE = *variate*	Standard errors of estimates
INVERSE = *symmetric matrix*	Inverse matrix
VCOVARIANCE = *symmetric matrix*	Variance-covariance matrix of parameters
DEVIANCE = *scalar*	Residual deviance
DF = *scalar*	Residual degrees of freedom
MVESTIMATES = *variate*	Estimates of missing values in series
SEMV = *variate*	Standard errors of estimates of missing values
COMPONENTS = *pointer*	Variates to save components of output series
SCORES = *variate*	To save scores (derivatives of the log-likelihood with respect to the parameters)

Description

An TFIT statement produces many quantities that you may want to use to assess, interpret and apply the fitted model. The TKEEP directive allows you to copy these quantities into Genstat data structures. If the METHOD option of the TFIT statement was set to initialize, then the results saved by the options SE, INVERSE, VCOVARIANCE and SCORE are unavailable. However, you can save the estimates of the missing values and their standard errors. The residual degrees of freedom in this case does not make allowance for the number of parameters in the model, but does allow for the missing values that have been estimated.

The OUTPUTSERIES parameter specifies the variate that was supplied by the SERIES parameter of the TFIT statement; this can be omitted.

You can use the RESIDUALS parameter to save the residuals in a variate, exactly as in the TFIT directive.

The ESTIMATES parameter can supply a variate to store the estimated parameters of the TSM. Each estimated parameter is represented once, but the innovation variance is omitted entirely. Genstat includes only the first of any set of parameters constrained to be equal using the FIX option of TFIT. The order of the parameters otherwise corresponds to their order in the variate of parameters in TSM, and is unaffected by any numbering used in the FIX option.

The SE parameter allows you to specify a variate to save the standard errors of the estimated parameters of the TSM. The values correspond exactly to those in the ESTIMATES variate. Parameters in a time series model may be aliased. This is detected when the equations for the estimates are being solved, and the message ALIASED is printed instead of the standard error when the PRINT option of TFIT or TDISPLAY includes the setting estimates. The corresponding units of the SE variate are set to missing values.

The INVERSE parameter can provide a symmetric matrix to save the product $(X'X)^{-1}$, where X is the most recent design matrix derived from the linearized least-squares regressions that were used to minimize the deviance. The ordering of the rows and columns corresponds exactly to that used for the ESTIMATES variate. The row of this matrix corresponding to any aliased parameter is set to zero except that the diagonal element is set to the missing value.

The VCOVARIANCE parameter allows you to supply a symmetric matrix for the estimated

variance-covariance matrix, $\hat{\sigma}_a^2(X'X)^{-1}$, of the TSM parameters. The ordering of the rows and columns and the treatment of aliased parameters corresponds exactly to that used for the ESTIMATES variate.

The DEVIANCE parameter specifies a scalar to hold the final value of the deviance criterion defined by the LIKELIHOOD option of TFIT.

The DF parameter saves the residual number of degrees of freedom, defined for a simple ARIMA model by $N-d-$(number of estimated parameters). If a seasonal model is used, this number is further reduced by Ds.

The MVESTIMATES parameter specifies a variate to hold estimates of the missing values of the series, in the order they appear in the series. You can thereby obtain forecasts of the series, by extending the SERIES in TFIT with a set of missing values. This is less efficient than using the TFORECAST directive, but it does have the advantage that the standard errors of the estimates take into account the finite extent of the data, and also the fact that the model parameters are estimated.

The SEMV parameter can supply a variate to hold the estimated standard errors of the missing values of the series, in the order they appear in the series.

The COMPONENTS parameter can be used after a multi-input model has been fitted using TFIT to access the components of the output series that are due to the various input series; you can also access the output noise. In simple regression, the input components are proportional to the input series. But the component resulting from a transfer-function model may be quite different from this. You can examine these components separately, or sum them to show the total fit to the output series that is explained by the input series. Note that the fitted values may appear to be offset from that output series, because the constant term is part of the noise component, and so is not included. You may want to examine the output noise component. For example, if you thought that the ARIMA model for the output noise was inadequate, you could investigate the noise component with univariate ARIMA modelling.

The SCORE parameter can specify a variate to hold the model scores. The scores are usually defined as the first derivatives of the log likelihood with respect to the model parameters. To get these, the scores supplied by TKEEP should be scaled by dividing by the estimated residual variance and reversing its sign. The elements of the SCORE variate correspond exactly to the parameters as they appear in the ESTIMATES variate. After using TFIT to fit a time series model, the scores should in theory be zero provided the model parameters do not lie on the boundary of their allowed range. The scores are used within TFIT to calculate the parameter changes at each iteration.

You can use the SAVE option to specify the time-series save structure from which the output is to be taken. By default TKEEP uses the structure from the most recent TFIT statement.

Option: SAVE.
Parameters: OUTPUTSERIES, RESIDUALS, ESTIMATES, SE, INVERSE, VCOVARIANCE, DEVIANCE, DF, MVESTIMATES, SEMV, COMPONENTS, SCORES.

See also
Directives: TSM, FTSM, TDISPLAY, TFILTER, TFIT, TFORECAST, TRANSFERFUNCTION, TSUMMARIZE.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

TRANSFERFUNCTION

Specifies input series and transfer-function models for subsequent estimation of a model for an output series.

Option

SAVE = *identifier*	To name time-series save structure; default *

Parameters

SERIES = *variates*	Input time series
TRANSFERFUNCTION = *TSMs*	Transfer-function models; if omitted, model with 1 moving-average parameter, lag 0
BOXCOXMETHOD = *string tokens*	How to treat transformation parameters (fix, estimate); default fix
PRIORMETHOD = *string tokens*	How to treat prior values (fix, estimate); default fix
ARIMA = *TSMs*	ARIMA models for input series

Description

TRANSFERFUNCTION can be used to define input series and transfer-function models to be used by subsequent TFIT statements.

In its simplest form, when the TRANSFERFUNCTION and PRIORMETHOD parameters are unset, TRANSFERFUNCTION can be used to specify the explanatory variables for a regression with autocorrelated errors.

The first parameter, SERIES, specifies a list of variates holding the time series of explanatory variables.

The BOXCOXMETHOD parameter allows you to estimate separate power transformations for the explanatory variables: the variable x_t is transformed to

$$x_t^{(\lambda)} = (x_t^{\lambda} - 1) / \lambda , \qquad \lambda \neq 0$$
$$x_t^{(0)} = \log(x_t)$$

The default is no transformation, corresponding to $x_t^{(\lambda)} = x_t$. You can choose whether the transformations are to be fixed or estimated, by specifying one string for each explanatory variable.

The ARIMA parameter allows you to associate with each explanatory variable a univariate ARIMA model for the time-series structure of that variable. If you think such a model is inappropriate, then you should give a missing value in place of the TSM identifier, or leave this parameter unset. You can use these models in any subsequent TFORECAST statement to incorporate, into the error limits of the forecasts, an allowance for uncertainties in the predicted explanatory variables; the allowance assumes that the future values of the explanatory variables are forecasts obtained using these ARIMA models.

The TRANSFERFUNCTION and PRIORMETHOD parameters are used to define multi-input transfer-function models.

The TRANSFERFUNCTION parameter specifies the transfer-function TSMs that are to be associated with the input series. A missing value in place of a TSM identifier causes Genstat to treat the corresponding input series as a simple explanatory variable, equivalent to a transfer-function model with orders (0,0,0,0).

The PRIORMETHOD parameter specifies, for each input series, how Genstat is to treat the transients associated with the early values of the transfer-function response. In calculating the input component z_t from the input x_t, Genstat has to make assumptions about the unknown values of x_t which came before the observation period. The default is that x_t (or generally $x_t^{(\lambda)}$) is assumed to be equal to the reference constant c of the transfer-function model. The pattern of the transient can be controlled by introducing a number $\max(p+d,b+q)$ of nuisance parameters

to represent the combined effects of all earlier input values on the observed output. Setting PRIORMETHOD=estimate specifies that these nuisance parameters are estimated so as to minimize the transients. You should, however, be careful in using this. Often all you will have to do is make a sensible choice of the reference constant *c*. Estimating the transients is best done as a final stage in refining the model; earlier, this may give poor numerical conditioning.

The SAVE option allows you to name the time-series save structure created by TRANSFERFUNCTION. You can use this identifier in a later TFIT statement, and eventually in a TFORECAST statement. If you do not name the save structure Genstat will use the most recent save structure, which will be overwritten each time a new TRANSFERFUNCTION statement is given.

Option: SAVE.
Parameters: SERIES, TRANSFERFUNCTION, BOXCOXMETHOD, PRIORMETHOD, ARIMA.

See also
Directives: TSM, FTSM, TDISPLAY, TFILTER, TFIT, TFORECAST, TKEEP, TSUMMARIZE.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

TREATMENTSTRUCTURE

Specifies the treatment terms to be fitted by subsequent ANOVA statements.

No options

Parameter

formula	Treatment formula, specifies the treatment model terms to be fitted by subsequent ANOVAs

Description

The TREATMENTSTRUCTURE directive defines the *treatment formula* which specifies treatment, or systematic, terms to be fitted in subsequent ANOVA statements. For a simple one-way analysis of variance this has the form

```
TREATMENTSTRUCTURE Tfac
```

where Tfac is a factor which indicates which treatment was received by each unit in the design. Most experiments, however, are devised to study several treatment factors. For these *factorial* experiments TREATMENTSTRUCTURE specifies a *model formula* to define the model terms to be fitted. Each model term will then have its own line in the analysis-of-variance table and, for example, will have a table of means.

Initially in a job, there is no treatment formula. This situation can be restored by a TREATMENTSTRUCTURE directive with a null setting:

```
TREATMENTSTRUCTURE
```

In its simplest form, a model formula is a list of *model terms*, linked by the operator "+". For example,

```
A + B
```

is a formula containing two terms, A and B, representing the main effects of factors A and B respectively. *Higher-order terms* (like interactions) are specified as series of factors separated by dots, but their precise meaning depends on which other terms the formula contains, as we explain below. The other operators provide ways of specifying a formula more succinctly, and of representing its structure more clearly.

The *crossing operator* * is used to specify factorial structures. For example, the treatment formula

```
TREATMENTSTRUCTURE Nitrogen * Sulphur
```

is expanded automatically by Genstat to become the formula

```
Nitrogen + Sulphur + Nitrogen.Sulphur
```

which has three terms: Nitrogen for the nitrogen main effect, Sulphur for the main effect of sulphur, and Nitrogen.Sulphur for the nitrogen by sulphur interaction. Higher-order terms like Nitrogen.Sulphur represent all the joint effects of the factors Nitrogen and Sulphur that have not been removed by earlier terms in the formula. Thus here it represents the interaction between nitrogen and sulphur as both main effects have been removed.

The other most-commonly used operator is the *nesting operator* (/). This occurs most often in block models (specified by the BLOCKSTRUCTURE directive). For example, the formula

```
Litter / Rat
```

is expanded to become the formula

```
Litter + Litter.Rat
```

This could define the block model for a design in which there are several litters of rats. As the formula contains no "main effect" for rat, the term Litter.Rat represents *rat-within-litter* effects (that is the differences between individual rats after removing any overall similarity between rats that belong to the same litter).

A formula can contain more than one of these operators. The three-factor factorial model

```
A * B * C
```

becomes

```
A + B + C + A.B + A.C + B.C + A.B.C
```

and the nested structure

```
Block / Wplot / Subplot
```

which specifies the block model of a split-plot design becomes

```
Block + Block.Wplot + Block.Wplot.Subplot
```

The operators can also be mixed in the same formula. In general, if *l* and *m* are two model formulae:

```
l * m  =  l + m + l.m
l / m  =  l + fac(l).m
```

(where *l*`.`*m* is the sum of all pairwise dot products of a term in *l* and a term in *m*, and `fac(`*l*`)` is the dot product of all factors in *l*). For example:

```
(A + B) * (C + D) = (A + B) + (C + D) + (A + B).(C + D)
                  = A + B + C + D + A.C + A.D + B.C + B.D
(A + B)/C = A + B + fac(A + B).C  =  A + B + A.B.C
```

The other important operator for `ANOVA` is the pseudo-factorial operator `//`. This allows you to partition an unbalanced treatment term into pseudo-terms, which are each balanced.

Contrasts can be fitted by putting a *function* of a factor into the treatment formula, instead of the factor itself. Polynomial contrasts can be specified using the `POL` or `POLND` functions. Other, user-defined regression models can be defined using the `REG` or `REGND` functions. `COMPARISON`, the other function relevant to `ANOVA` allows comparisons to be calculated between levels of the factor.

Options: none.
Parameter: unnamed.

See also
Directives: `ANOVA`, `BLOCKSTRUCTURE`, `COVARIATE`, `ADISPLAY`, `AKEEP`.
Procedures: `AFCOVARIATES`, `ASTATUS`, `AUNBALANCED`.
Functions: `COMPARISON`, `POL`, `POLND`, `REG`, `REGND`.
Genstat Reference Manual 1 Summary section on: Analysis of variance.

TREE

Declares one or more tree data structures and initializes each one to have a single node known as its root.

No options

Parameter

IDENTIFIER = *identifiers*	Identifiers of the trees

Description

TREE declares and initializes Genstat tree structures. These can be used to represent hierarchical structures like classification trees, identification keys and regression trees. These types of tree can be constructed by special-purpose procedures BCLASSIFICATION, BKEY and BREGRESSION, respectively, and displayed by procedures BGRAPH and BPRINT. Most users will use only these special-purpose procedures, and will not need to operate on trees directly, nor to be aware of how they are formed, stored or manipulated. The procedures, however, are based on a suite of directives, functions and procedures summarized below, which provide the tool kit not only for the officially-supported tree facilities but also for user enhancements and extensions.

The tree structure is like a real tree, which starts from a root and then splits into branches, except that it is usually viewed as growing downwards instead of upwards. The branch-points in the tree are known as *nodes*, with the initial node being called the *root* (as in a real tree). There is also a node at the end of each branch, known as its *terminal node*. In Genstat a tree is similar to a pointer, with an element for each node. These elements are the identifiers of data structures which can be used to store information about the nodes. Usually the data structures will be pointers, so that several pieces of information can be stored for each node, but the precise contents depend on the type of tree (see, for example, procedures BCLASSIFICATION, BKEY and BREGRESSION).

Each node thus has a number, corresponding to the index of its element in the tree. The root is always numbered one, and this is the only node that the tree contains when it is declared by TREE. Further nodes can be added by the BGROW or BJOIN directives, which form branches from a terminal node or join another tree to a terminal node, respectively. The converse process of cutting a tree at a defined node and discarding the nodes and information below it is provided by the BCUT directive.

The numbers of the subsequent nodes can be obtained from the functions that are provided to navigate around a tree:

BNEXT	provides the numbers of the nodes below a node;
BPREVIOUS	provides the number of the node immediately above a node;
BTERMINAL	finds the next terminal node after a node;
BSCAN	finds the number of the node immediately after a node in a standard branch-by-branch order that visits each node once.

Other useful functions include:

BNBRANCHES	provides the number of branches below a node;
BDEPTH	calculates the depth of a node (taking the root as being at depth 1);
BPATH	provides a variate containing the numbers of the nodes on the branch to a node;
BBRANCHES	provides a variate containing the numbers of the branches taken on the path to a node;
BBELOW	provides a variate containing numbers of all the nodes or

	all the terminal nodes below a node;
BNNODES	provides the number of nodes in a tree;
BMAXNODE	provides the maximum node number in a tree.

There are also several utility procedures, which are used by the special-purpose tree procedures.

BCONSTRUCT	constructs a tree (using subsidiary procedure BSELECT, which is customized according to the type of tree).
BGRAPH	plots a tree.
BPRINT	displays a tree.
BPRUNE	prunes a tree using minimal cost complexity (assuming that "accuracy" values have been stored at each node of the tree, which can be done using customized procedure BVALUES).

New tree-based analyses can thus be added by writing a main procedure (like BCLASSIFICATION, BREGRESSION etc), and defining appropriate versions of BSELECT.

Options: none.
Parameter: IDENTIFIER.

See also
Directives: BASSESS, BCUT, BGROW, BJOIN, POINTER.
Procedures: BCONSTRUCT, BCLASSIFICATION, BGRAPH, BKEY, BPRINT, BPRUNE.
Functions: BBELOW, BBRANCHES, BDEPTH, BMAXNODE, BNBRANCHES, BNEXT, BNNODES, BPATH, BPREVIOUS, BSCAN, BTERMINAL.
Genstat Reference Manual 1 Summary section on: Data structures.

TRY

Displays results of single-term changes to a linear, generalized linear or generalized additive model.

Options

PRINT = *string tokens*	What to print (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, changes, confidence); default chan
FACTORIAL = *scalar*	Limit for expansion of model terms; default * i.e. that in previous TERMS statement
POOL = *string token*	Whether to pool ss in accumulated summary between all terms fitted in a linear model (yes, no); default no
DENOMINATOR = *string token*	Whether to base ratios in accumulated summary on rms from model with smallest residual ss or smallest residual ms (ss, ms); default ss
NOMESSAGE = *string tokens*	Which warning messages to suppress (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation); default *
FPROBABILITY = *string token*	Printing of probabilities for variance and deviance ratios (yes, no); default no
TPROBABILITY = *string token*	Printing of probabilities for t-statistics (yes, no); default no
SELECTION = *string tokens*	Statistics to be displayed in the summary of analysis produced by PRINT=summary, seobservations is relevant only for a Normally distributed response, and %cv only for a gamma-distributed response (%variance, %ss, adjustedr2, r2, seobservations, dispersion, %cv, %meandeviance, %deviance, aic, bic, sic); default %var, seob if DIST=normal, %cv if DIST=gamma, and disp for other distributions
PROBABILITY = *scalar*	Probability level for confidence intervals for parameter estimates; default 0.95

Parameter

formula	List of explanatory variates and factors, or model formula

Description

TRY investigates modifications to the current regression model, which may be linear, generalized linear or generalized additive. Terms in the specified formula are dropped from the current model if they are already there, or are added to it if they are not. It is best to give a TERMS statement before using TRY to define a common set of units for the models to be investigated. If no model is fitted after the TERMS statement, the current model is taken to be the null model.

The default setting, changes, of the PRINT option summarises the effects of the changes after they have all been tried. Other settings request further details of the changed models. These are printed after each change. Genstat then restores the original model before trying the next change.

The options of TRY are otherwise the same as those of the FIT directive, except that there is no CONSTANT option. The accumulated setting of the PRINT option will show only one change at a time. Accumulated summaries produced by later statements will not have any entries for a

TRY statement.

Options: PRINT, FACTORIAL, POOL, DENOMINATOR, NOMESSAGE, FPROBABILITY, TPROBABILITY, SELECTION, PROBABILITY.
Parameter: unnamed.

Action with RESTRICT
If a TERMS statement was given before fitting the model, any restrictions on the variates or factors in the model will have been implemented then. So any new restrictions on vectors involved in the model specified by TRY will be ignored. If no TERMS statement has been given and TRY involves new terms not already in the model, restrictions on the variates or factors in these terms will be taken into account and may cause the units involved in the regression to be redefined.

See also
Directives: MODEL, TERMS, FIT, ADD, DROP, SWITCH, STEP.
Procedures: RSCREEN, RSEARCH.
Genstat Reference Manual 1 Summary section on: Regression analysis.

TSM

Declares one or more TSM data structures.

Option

MODELTYPE = *string token*	Type of model (arima, transfer); default arim

Parameters

IDENTIFIER = *identifiers*	Identifiers of the TSMs
ORDERS = *variates*	Orders of the autoregressive, integrated and moving-average parts of each TSM
PARAMETERS = *variates*	Parameters of each TSM
LAGS = *variates*	Lags, if not default

Description

The TSM structure stores a time-series model which you can use with directives such as TFIT for Box-Jenkins modelling of time series. The information that you give to specify the model is stored in two variates, called the *orders* and the *parameters*; an optional third variate contains *lags*. The elements of a TSM are thus

[1] or ['Orders'];
[2] or ['Parameters'];
[3] or ['Lags'].

The labels of the TSM can be specified in either upper or lower case, or any mixture.

To declare a TSM you use the TSM directive. You set the type of model by the MODELTYPE option. The default setting defines an *ARIMA model*. This is an equation relating the present value y_t of an observed time series to past values. The equation includes lagged values not only of the series itself, but also of an unobserved series of *innovations*, a_t; you can interpret the innovations as the error in predicting y_t from past values y_{t-1}, y_{t-2} .... The usual statistical model assumes that the innovations are a series of independent Normal deviates with mean zero and constant variance. The residuals obtained from fitting the model can be used to estimate the innovations.

Using the notation of Box & Jenkins (1970), the simple non-seasonal ARIMA model for the time series y_t is

$$\varphi(B)\ \{\nabla^d y_t^{(\lambda)} - c\} = \theta(B)a_t$$

where B is the backward shift operator $B^p y_t = y_{t-p}$,
∇ is the differencing operator $\nabla y_t = y_t - y_{t-1}$, $\nabla^d y_t = \nabla^{d-1}(y_t - y_{t-1})$, and

$$\varphi(B) = 1 - \varphi_1 B - ... - \varphi_p B^p$$
$$\theta(B) = 1 - \theta_1 B - ... - \theta_q B^q$$

The parameter λ specifies a Box-Cox power transformation defined by

$$y_t^{(\lambda)} = (y_t^{\lambda} - 1) / \lambda, \quad \lambda \neq 0$$
$$y_t^{(0)} = \log(y_t)$$

However, in the default case when λ is fixed and not estimated, the value λ=1 implies no transformation and then $y_t^{(1)} = y_t$ rather than $y_t - 1$. If $\lambda \neq 1$ or if λ is to be estimated, then Genstat will not let you have values of $y_t \leq 0$. The usual case however is that λ=1 and is not to be estimated, so that y_t may take any values.

The ORDERS parameter is a list of variates, one for each of the models. For each simple ARIMA model, the variate contains the three values p, d and q.

The PARAMETERS parameter is a list of variates, one for each of the models. For each simple ARIMA model, the variate contains $(3+p+q)$ values: λ, c, σ_a^2, $\varphi_1 ... \varphi_p$, $\theta_1 ... \theta_q$. You must always include the first three parameters. The parameter σ_a^2 is the innovation variance.

Whenever a TSM is used, Genstat checks its values. The orders must all be non-negative. The parameters λ and c can take any values, but σ_a^2 must be non-negative. The next $p+q$ values

specify the autoregressive and moving-average parameters: they must satisfy the stationarity and invertibility conditions for ARIMA models (see Box & Jenkins 1970). An exception is that before estimation the model parameters may be unset, in which case Genstat sets them to default values. You can omit the PARAMETERS parameter, in which case an unnamed structure is defined to contain the default values. However, you should usually specify the variate of parameters, and if possible assign good preliminary values before estimation (see FTSM) as this will speed up the model fitting process.

The LAGS parameter is a list of variates, one for each of the models. For each simple ARIMA model, this variate contains $p+q$ values, one corresponding to each of the autoregressive and moving-average parameters. Genstat then modifies the ARIMA model by defining

$$\varphi(B) = 1 - \varphi_1 B{**}l_1 - ... - \varphi_p B{**}l_p$$
$$\theta(B) = 1 - \theta_1 B{**}m_1 - ... - \theta_q B{**}m_q$$

The LAGS parameter for this model contains $l_1...l_p, m_1...m_q$. The sequences of lags $l_1...l_p$ must be positive integers that are strictly increasing; the default values are $1...p$ if LAGS is not set. The same rule applies to $m_1...m_q$.

The seasonal ARIMA model for the time series y_t is an extension of the simple model, to the form

$$\varphi(B)\,\Phi(B^s)\,\{\,\nabla^d\nabla_s^{\,D} y_t^{(\lambda)} - c\,\} = \theta(B)\,\Theta(B^s)\,a_t$$

where the extra, seasonal, operators associated with seasonal period s are of three types:

$$\Phi(B^s) = 1 - \Phi_1 B^s - ... - \Phi_P B{**}P_s$$

which is seasonal autoregression of order P;

$$\nabla_s^{\,D}$$

which is seasonal differencing of order D; and

$$\Theta(B^s) = 1 - \Theta_1 B^s - ... - \Theta_Q B{**}Q_s$$

which is seasonal moving average of order Q.

When seasonal terms are to be included, you must extend the ORDERS parameter so that it contains p, d, q, P, D, Q and s. Even if the non-seasonal part of the model has $p=d=q=0$, these parameters must still be included at the beginning of the list. The seasonal orders must satisfy $P\geq 0$, $D\geq 0$, $Q\geq 0$ and $s\geq 1$.

You must also extend the PARAMETERS parameter to contain:

$$\lambda,\ c,\ \sigma_a^2,\ \varphi_1...\varphi_p,\ \theta_1...\theta_q,\ \Phi_1...\Phi_P,\ \Theta_1...\Theta_Q$$

You can modify the seasonal model to allow other lags:

$$\Phi(B^s) = 1 - \Phi_1 B{**}L_1 - ... - \Phi_P B{**}L_p$$
$$\Theta(B^s) = 1 - \Theta_1 B{**}M_1 - ... - \Theta_Q B{**}M_Q$$

The sequence of lags $L_1...L_P$ must be strictly increasing and must be positive-integer multiples of the period s; the default values are s, $2s$ ... Ps. The same rules apply to $M_1...M_Q$. For any seasonal model, you must extend the LAGS parameter, if supplied, so that it contains

$$l_1 ... l_p, m_1 ... m_q, L_1 ... L_P, M_1 ... M_Q.$$

You can use multiple seasonal periods, by extending the variate of ORDERS with further seasonal orders P', D', Q' and s'. You must correspondingly extend the variates of PARAMETERS and LAGS. It is also possible to set the seasonal periods to 1, which means you can estimate non-seasonal models with factored operators.

You can declare an ORDERS variate to have more values than is necessary, provided that the extra values are filled with zeroes, and that the number of values is $3+4k$, k being the number of seasonal periods. The same applies to PARAMETERS and LAGS variates, except that Genstat ignores the extra values whatever they may be. Thus you can extend a simple model to a seasonal model, simply by resetting the extra values.

Setting MODELTYPE=transferfunction defines a *transfer-function model*. The simple non-seasonal transfer-function model relates a component z_t of the output series to the corresponding input series x_t, by the equation

$$\delta(B)\,\nabla^d z_t = \omega(B)\,B^b\,\{x_t^{(\lambda)} - c\}$$

where

$$\delta(B) = 1 - \delta_1 B - \ldots - \delta_p B^P$$

$$\omega(\mathrm{B}) = \omega_0 - \omega_1 B - \ldots - \omega_q B^q .$$

The integer b>0 defines a pure *delay*, and the integer d>0 defines the order of differencing in the transfer function.

The parameter λ specifies a Box-Cox power transformation for the input series, and the parameter c specifies a reference level for the transformed input. There is no mean correction of the input series when transfer-function models are estimated, and you should use a value of c close to the series mean so as to improve the numerical conditioning of the estimation procedure. However, if the input series x_t is trend-like rather than stationary, you could alternatively use a value for c close to the early series values, because this reduces the transient errors that arise when the transfer function is applied. The PRIORMETHOD parameter of TRANSFERFUNCTION, described below, provides further means of handling these transients.

The parameters λ and c are not estimated unless you specify otherwise by the BOXCOXMETHOD parameter of TRANSFERFUNCTION or the FIX option of TFIT. Often c in the transfer-function model is aliased with the constant term in the ARIMA errors, and so they should not both be estimated. In some circumstances, however, they both could be estimated, for example in a differenced transfer-function model with stationary noise.

The ORDERS parameter for the simple transfer-function model described above specifies a variate containing the four values b, p, d and q.

The PARAMETERS parameter specifies a variate containing 3+p+q values: λ, c, δ_1, ... δ_p, ω_0, ω_1 ... ω_q. You must always include the parameters λ, c and ω_0. When you use a transfer-function model, Genstat will check its parameter values. In particular the operator $\delta(B)$ must satisfy the stability or stationarity condition.

The LAGS parameter is optional, and may be used to change the lags associated with the parameters, from the default values of 1 ... p, 1 ... q. The variate of lags contains values corresponding to the parameters δ_1 ... δ_p, ω_1 ... ω_q. They have the same interpretation as the lags in ARIMA models, and must satisfy the same conditions. Note that there is no lag associated with ω_0, because the delay b provides the necessary flexibility for this.

You can also have seasonal extensions of transfer-function models:

$$\delta(B)\Delta(B^s)\nabla^d\nabla_s^D z_t = \omega(B)\Omega(B^s)B^b\{x_t^{(\lambda)} - c\}$$

$$\Delta(B^s) = 1 - \Delta_1 B^s - \ldots - \Delta_P B^{Ps}$$

$$\Omega(B^s) = 1 - \Omega_1 B^s - \ldots - \Omega_Q B^{Qs}$$

Note that there is no Ω_0 coefficient, because ω_0 is always present in the model and provides sufficient flexibility.

The ORDERS parameter here contains b, p, d, q, P, D, Q and s, and the PARAMETERS parameter contains λ, c, δ_1 ... δ_p, ω_0 ... ω_q, Δ_1 ... Δ_P, Ω_1 ... Ω_Q. You can analogously extend the LAGS parameter. You can also have extensions to multiple seasonal periods, as for ARIMA models.

Option: MODELTYPE.
Parameters: IDENTIFIER, ORDERS, PARAMETERS, LAGS.

Reference

Box, G.E.P. & Jenkins, G.M. (1970). *Time Series Analysis, Forecasting and Control*. Holden-Day, San Francisco.

See also

Directives: FTSM, TDISPLAY, TFILTER, TFIT, TFORECAST, TKEEP, TRANSFERFUNCTION, TSUMMARIZE, CORRELATE, FOURIER, POINTER.

Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY, MOVINGAVERAGE, PERIODTEST, PREWHITEN, REPPERIODOGRAM, SMOOTHSPECTRUM.

Genstat Reference Manual 1 Summary sections on: Data structures, Time series.

TSUMMARIZE

Displays characteristics of time series models.

Options

PRINT = *string tokens*	What to print (autocorrelations, expansion, impulse, piweight, psiweight); default *
GRAPH = *string tokens*	What to display with graphs (autocorrelations, impulse, piweight, psiweight); default *
MAXLAG = *scalar*	Maximum lag for results; default 30

Parameters

TSM = *TSMs*	Models to be displayed
AUTOCORRELATIONS = *variates*	To save theoretical autocorrelations
IMPULSERESPONSE = *variates*	To save impulse-response function
STEPFUNCTION = *variates*	To save step function from impulse
PIWEIGHTS = *variates*	To save pi-weights
PSIWEIGHTS = *variates*	To save psi-weights
EXPANSION = *TSMs*	To save expanded models
VARIANCE = *scalars*	To save variance of each TSM

Description

The TSUMMARIZE directive helps you investigate time-series models by displaying or saving various characteristics. These are the theoretical autocorrelation function of an ARIMA model, and the pi-weights and psi-weights; also the impulse-response function of a transfer-function model. TSUMMARIZE can derive the expanded form of a model, in which all seasonal terms are combined with the non-seasonal term.

For an ARIMA model in the TSM parameter, you can set only the AUTOCORRELATIONS, PSIWEIGHTS and PIWEIGHTS parameters. Also, you can set the IMPULSERESPONSE parameter only for a transfer-function model. You can set the EXPAND parameter for either type of model. The TSMs in any TSUMMARIZE statement must be completely defined; that is, you must have set the orders and parameters, and the lags if you are using them. The only exceptions are that Genstat takes the transformation parameter to be 1.0 if it is missing, and that the innovation variance of an ARIMA model need not be set.

The MAXLAG option specifies the maximum lag to which Genstat is to do calculations: this applies to autocorrelations, psi-weights, pi-weights and impulse responses. If MAXLAG is unset, the maximum lag is defined implicitly as the length of the first variate in the parameters. However, if the length of this variate is also undefined, the maximum lag cannot be defined and Genstat reports a fault.

You can set the PRINT and GRAPH options independently of the parameters: these store results, and display the various characteristics of models.

The AUTOCORRELATIONS parameter allows you to store the theoretical autocorrelation function of an ARIMA model. Such a model uniquely defines an autocorrelation function whose values $r_0 \ldots r_m$ are assigned by Genstat to the variate R, where m is the maximum lag. If the model has differencing parameters d=D=0, then the autocorrelation function is that of a series y_t that follows this model.

If either d>0 or D>0, then the theoretical autocorrelations are calculated as if d=D=0, and so they correspond to those of the differenced y_t series. This is because the autocorrelations of y_t are undefined for non-stationary models.

The PSIWEIGHTS parameter allows you to store the theoretical psi-weights $\psi_0 \ldots \psi_m$ of an ARIMA model. These are used internally by Genstat when error limits are calculated for forecasts obtained using the model. You will need them for example if you want to calculate the

variance of the total of the forecast values up to some specified maximum lead time. They are defined for a non-seasonal model by

$$1 + \psi_1 B + \psi_2 B^2 + ... = \theta(B) / \{ \varphi(B)\nabla^d \}$$

The PIWEIGHTS parameter allows you to store the theoretical pi-weights $\pi_0 ... \pi_m$ of an ARIMA model: these show explicitly how past values contribute to a forecast. The weights are defined by:

$$1 - \pi_1 B - \pi_2 B^2 - ... = \{ \varphi(B)\nabla^d \} / \theta(B)$$

The IMPULSERESPONSE parameter allows you to store the theoretical impulse-response function, $v_0 ... v_m$, of a transfer-function model. This function can help you interpret the model. The sequence is defined for a non-seasonal transfer-function model by:

$$\nu_0 + \nu_1 B + \nu_2 B^2 + ... = \omega(B)B^b / \{ \delta(B)\nabla^d \}$$

For an ARIMA model you can combine into one generalized autoregressive operator all the differencing operators, the non-seasonal autoregressive operators, and the seasonal autoregressive operators. The non-seasonal and seasonal moving-average operators may similarly be combined. This expanded model can be printed using the expansion setting of PRINT and saved using the EXPANSION parameter. It can be used to help you understand a series. But you might also want to re-estimate the parameters in the expanded model, to test whether the differencing operators or seasonal factors unnecessarily constrain the structure of the original model. If you have not previously defined one of the identifiers supplied by the EXPANSION parameter, Genstat will automatically define it to be a TSM, and its component variates will be set up to have the length defined by the corresponding model in the TSM parameter. The expansion does not change the transformation parameter of the model, nor the constant term, nor the innovation variance. If the model that you have supplied contains non-zero differencing orders, then the generalized model does not satisfy the stationarity constraint on the parameters; neither does the constant term have the same interpretation as it had in the supplied model. The expansion of transfer-function models exactly parallels that of ARIMA models.

Options: PRINT, GRAPH, MAXLAG.
Parameters: TSM, AUTOCORRELATIONS, IMPULSERESPONSE, STEPFUNCTION, PIWEIGHTS, PSIWEIGHTS, EXPANSION, VARIANCE.

See also

Directives: TSM, FTSM, TDISPLAY, TFILTER, TFIT, TFORECAST, TKEEP, TRANSFERFUNCTION, CORRELATE.
Procedures: BJESTIMATE, BJFORECAST, BJIDENTIFY.
Genstat Reference Manual 1 Summary section on: Time series.

TXBREAK

Breaks up a text structure into individual words.

Option

SEPARATOR = *text*	Defines the characters separating the words in the original text; default `' ,;:.'`

Parameters

TEXT = *texts*	Text to break into words
WORDS = *texts*	Saves the words contained in each text (in the order in which they occur)
COLUMNS = *variates*	Saves the number of the column in the TEXT where each word began
LINES = *variates*	Saves the number of the line where each word was found
PLACESINLINES = *variates*	Saves the place of each word (first, second &c) within the line where it was found

Description

The TXBREAK directive forms a text containing all the words (including duplicates) found in a text. The original text to break up is supplied by the TEXT parameter, and the WORDS parameter saves a text storing the words that it contains. The words are stored in the order in which they occur in the original text (but, for example, you could use the SORT directive to sort them into alphabetic order). The LINES parameter can save a variate recording the line in the original text where each one was found. The COLUMNS parameter can save a variate recording the column where each word began, and the PLACESINLINES parameter can save a variate giving the place of each word (first, second &c) within the line where it was found.

By default, the words are assumed to be separated from one another by spaces or by any of the standard punctuation characters (comma, semi-colon, colon, full stop). However, you can use the SEPARATOR option to specify some other characters. For example, you could put `SEPARATOR=' ,;:.?'` to allow question marks as well. These characters are all removed from the words when they are stored.

Option: SEPARATOR.
Parameters: TEXT, WORDS, COLUMNS, LINES, PLACESINLINES.

Action with RESTRICT

TXBREAK takes account of any restrictions on the original text, and omits the words in the restricted lines.

See also

Directives: TEXT, CONCATENATE, EDIT, TXCONSTRUCT, TXFIND, TXPOSITION, TXREPLACE.
Procedure: TXSPLIT.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

TXCONSTRUCT

Forms a text structure by appending or concatenating values of scalars, variates, texts, factors pointers or formulae; allows the case of letters to be changed or values to be truncated and reversed.

Options

TEXT = *text*	Stores the text that is formed
CASE = *string token*	Case to use for letters (given, lower, upper, changed, sentence, title); default give leaves the case of each letter as given in the original texts
METHOD = *string token*	Whether to append or concatenate the values of the structures (append, concatenate) default conc
SEPARATOR = *string*	Characters to separate all except last two strings in each line when concatenating; default '' (i.e. none)
LASTSEPARATOR = *string*	Characters to separate last two strings in each line when concatenating; default uses the characters defined by SEPARATOR
PREFIX = *string*	Characters to put at the start of each line when concatenating; default '' (i.e. none)
END = *string*	Characters to put at the end of each line when concatenating; default '' (i.e. none)
SIGNIFICANTFIGURES = *scalar*	Specifies the number of significant figures to include for numerical data; default 4

Parameters

STRUCTURE = *scalars*, *variates*, *factors*, *texts*, *pointers* or *formulae*	Structures whose values are to be appended or concatenated
WIDTH = *scalars* or *variates*	Number of characters to take from the strings formed from the units of each STRUCTURE, a negative value takes all the (unskipped) characters other than trailing spaces; if omitted or set to a missing value, all the (unskipped) characters are taken
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numerical structures; if omitted or set to a missing value, a default is used which aims to print the value to the precision defined by the SIGNIFICANTFIGURES option
SKIP = *scalars* or *variates*	Number of characters to skip at the left-hand side of the strings formed from the units of each STRUCTURE, a negative value skips all initial spaces; if omitted or set to a missing value, no characters are skipped
FREPRESENTATION = *string tokens*	How to represent factor values (labels, levels, ordinals); default is to use labels if available, otherwise levels
DREPRESENTATION = *scalars* or *texts*	Format to use for dates and times (stored in numerical structures)
REVERSE = *string tokens*	Whether to reverse the strings of characters formed from the units of each structure (yes, no); default no
MISSING = *texts*	String to use to represent missing values of numerical structures; default '*'

Description

The TXCONSTRUCT directive forms a text from the values of scalars, variates, texts, factors or pointers. The new text is saved using the TEXT option, and the structures from which it is to be formed are listed using the STRUCTURE parameter.

By default the values of the structures are concatenated alongside each other (as with the CONCATENATE directive); alternatively you can set option METHOD=append to append them below each other. When you are concatenating, the structures in the STRUCTURE list must generally contain the same number of values (and this then defines the number of lines in the new text). The exception is that the STRUCTURE list can include scalars or texts containing a single string if you want to put the same numbers or strings into every line of the new text.

Numerical values (from scalars, variates or factors) are converted into strings of characters before they are used. As in the PRINT directive, you can use the DREPRESENTATION parameter to indicate whether these are to be treated as dates. Alternatively, if they are to remain as numbers, the DECIMALS parameter specifies the number of decimal places to use. DECIMALS can be set to a scalar if all the values of the structure are to be printed with the same number of decimals, or to a variate if you want to represent different units of a variate or factor structure with different numbers of decimals. The SIGNIFICANTFIGURES option specifies the number of significant figures to aim for if DECIMALS is not set, or if it contains missing values (default 4). A numerical value will then be converted as though it had been printed with the number of decimals required to give SIGNIFICANTFIGURES significant figures, and any trailing zero decimal values had then been removed. Missing numerical values are represented by the asterisk character (*) by default, in the usual way, but you can specify another string of characters using the MISSING parameter.

A formula is converted to a text before being concatenated. The maximum width is defined as 200. So this will be a text with one line, unless the result is more than 200 characters wide.

The SKIP parameter allows you to skip characters at the start of the s trings provided by each structure. You can supply a scalar to skip the same number of characters in every string, or a variate if you want to make different skips in every string. Similarly the WIDTH parameter specifies how many characters are to be taken, after omitting any initial characters as specified by SKIP. The strings formed from scalars, variates, factors and pointers do not contain any initial or trailing spaces. You can set a negative skip to ignore all the initial spaces in a string taken from a text structure, and set a negative width to ignore all its trailing spaces. The REVERSE parameter allows you to reverse the strings from any of the structures.

The CASE option enables you to change the case of letters in the strings. The available settings are:

given	to leave the case of each letter exactly as given in the string;
upper	to change all letters to upper case (or capitals);
lower	to change all letters to lower case;
changed	to put lower-case letters into upper case, and upper-case letters into lower case;
sentence	to put the first character in the text (if a letter) into upper case, then to use upper case only at the start of each new sentence;
title	to begin each new word with a capital letter, but otherwise to use lower case.

When METHOD=concatenate you can use the SEPARATOR, LASTSEPARATOR, PREFIX and END options to insert characters automatically between the adjacent pairs of strings in each line. LASTSEPARATOR supplies a string of characters to insert between the last pair of strings, SEPARATOR supplies characters to insert between all the other pairs of strings, PREFIX supplies

characters to put at the start of each line, and END supplies characters to put at the end of each line. The defaults for SEPARATOR, PREFIX and END are the empty string '', while LASTSEPARATOR uses the characters defined by SEPARATOR as its default. So by default no characters are inserted.

Options: TEXT, CASE, METHOD, SEPARATOR, LASTSEPARATOR, PREFIX, END, SIGNIFICANTFIGURES.
Parameters: STRUCTURE, WIDTH, DECIMALS, SKIP, FREPRESENTATION, DREPRESENTATION, REVERSE, MISSING.

Action with RESTRICT

TXCONSTRUCT takes account of restrictions on any of the vectors that occur in the statement. If more than one vector is restricted, then each such restriction must be the same. The values of the units in the new text that are excluded by the restriction are left unchanged.

See also

Directives: TEXT, CONCATENATE, EDIT, EQUATE, TXBREAK, TXFIND, TXPOSITION, TXREPLACE.
Procedures: APPEND, FVSTRING, SUBSET, STACK, TXPROGRESSION, UNSTACK.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

TXFIND

Finds a subtext within a text structure.

Options

CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for the SUBTEXT within the TEXT (significant, ignored); default sign
REVERSE = *string token*	Whether to reverse the search to work from the end of the TEXT (yes, no); default no
MULTISPACES = *string token*	Whether to treat differences between multiple spaces and single spaces as significant, or to treat them all like a single space (significant, ignored); default sign
DISTINCT = *string tokens*	Whether to require the SUBTEXT to have one or more separators to its left or right within the TEXT (left, right); default *
SEPARATOR = *string*	Characters to use as separators; default ' ,;:.'
SAMELINE = *string token*	Whether to ignore matches in the TEXT where the SUBTEXT is not all on the same line (yes, no); default no

Parameters

TEXT = *texts*	Texts to be searched
SUBTEXT = *texts*	Text to look for in each TEXT
COLUMN = *scalars*	Position of the column within TEXT where the first character of SUBTEXT has been found
LINE = *scalars*	Number of the line within TEXT where the first character of SUBTEXT has been found
ICOLUMN = *scalars*	Column within TEXT at which to start the search
ILINE = *scalars*	Line within TEXT at which to start the search
ENDCOLUMN = *scalars*	Position of the column within TEXT where the last character of SUBTEXT has been found
ENDLINE = *scalars*	Number of the line within TEXT where the last character of SUBTEXT has been found

Description

The TXFIND directive looks for a Genstat text structure within another text structure. The text to search is specified by the TEXT parameter, and the SUBTEXT parameter specifies the text to be found. The search treats the two texts as if they were paragraphs of characters: that is, it takes no account of the line breaks within the two text structures, replacing each one with a space. The COLUMN parameter saves the column within the TEXT where the first character of the SUBTEXT is found, and the LINE parameter saves its line within the TEXT. These are both set to zero if SUBTEXT is not found. Similarly the ENDCOLUMN and ENDLINE parameters save the position of the last character of the SUBTEXT. You can use the ICOLUMN and ILINE parameters to specify a starting column and line for the search. So you can search for the next occurrence of SUBTEXT by setting ILINE to the saved value of LINE, and ICOLUMN to the saved value of COLUMN plus one.

TXFIND usually takes account of the case of letters (small or capital) when looking for the SUBTEXT within the TEXT. So for example 'Genstat' would not match with 'Genstat'. However, you can set option CASE=ignored to ignore differences in case. It will usually also treat multiple spaces as significant, but you can set option MULTISPACE=ignored to treat them all like a single space.

Option DISTINCT is useful if you are looking for distinct words or phrases. The left setting requires the SUBTEXT to begin either at the start of the TEXT, or to be preceded in the TEXT by a separator (such as a space or comma). Similarly, the right setting requires the SUBTEXT to end within the TEXT with a separator (or to be at the end of the TEXT). The separators are specified by the SEPARATOR option.

By default, the SUBTEXT can be split over several lines of the TEXT, but you can set option SAMELINE=yes to ensure that it will be recognised only if it is all on a single line.

Options: CASE, REVERSE, MULTISPACES, DISTINCT, SEPARATOR, SAMELINE.
Parameters: TEXT, SUBTEXT, COLUMN, LINE, ICOLUMN, ILINE, ENDCOLUMN, ENDLINE.

Action with RESTRICT
Any restrictions are ignored.

See also
Directives: TEXT, CONCATENATE, EDIT, GETLOCATIONS, TXBREAK, TXCONSTRUCT, TXPOSITION, TXREPLACE.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

TXINTEGERCODES

Converts textual characters to and from their corresponding integer codes.

Options

CONVERTTO = *string token*	Whether to convert from text characters to integer codes or integer codes to text characters (codes, text) ; default code
REPRESENT = *string token*	How to treat code values 128-255 (extendedascii, utf8); default exte if CODES defines no characters that can be represented only in UTF-8, otherwise utf8

Parameters

TEXT = *texts*	Text structures (each with a single line only)
CODES = *variates* or *scalars*	Integer codes corresponding to the characters in each text

Description

Textual characters all have corresponding integer code values (see http://unicode.org/charts/). For example, the characters in the basic ASCII character set have codes running from 0 to 127. The letters a-z have codes 97-122, the capital letters have codes 65-90, and the digits 0-9 have codes 48-57. These characters can all be represented by a single "byte" of computer storage, consisting of eight "bits" each able to store either one or zero. Genstat stores other characters, such as those in the Chinese, Korean or Thai languages, in the UTF-8 format which uses up to four bytes per character.

By default, TXINTEGERCODES takes as input a text supplied by the TEXT parameter, which must contain only one line. The codes corresponding to the characters in the line are saved in a variate, supplied by the CODES parameter. Alternatively, if you set option CONVERTTO = text, the codes are taken as input, and TEXT saves the corresponding line of characters. Missing or zero codes are ignored, and invalid codes (for example, negative numbers) are faulted.

Codes 128-255 can be represented either by characters in the extended ASCII character set, or by 2-byte UTF-8 characters. These represent the same actual characters, but you may find one representation more convenient than the other, depending on how you want to use any output involving the text in future. If you have a preference, you can control this by setting the REPRESENT option. Otherwise, TXINTEGERCODES uses extended ASCII characters, unless the variate contains codes that can be represented only in UTF-8.

Options: CONVERTTO, REPRESENT.
Parameters: TEXT, CODES.

Action with RESTRICT

TXINTEGERCODES ignores any restrictions on the parameters.

See also

Directive: TEXT.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

TXPOSITION

Locates strings within the lines of a text structure.

Options

CASE = *string token*	Whether to treat the case of letters as significant when searching for lines of the SUBTEXT within the TEXT (significant, ignored); default sign
REVERSE = *string token*	Whether to reverse the search to work from the end of the lines of the TEXT (yes, no); default no
MULTISPACES = *string token*	Whether to treat differences between multiple spaces and single spaces as significant, or to treat them all like a single space (significant, ignored); default sign
DISTINCT = *string tokens*	Whether to require the SUBTEXT to have one or more separators to its left or right within the TEXT (left, right); default *
SEPARATOR = *text*	Characters to use as separators; default ' ,;:.'

Parameters

TEXT = *texts*	Texts whose strings are to be searched
SUBTEXT = *texts*	Specifies a string or strings to find in each TEXT
POSITION = *variates*	Position of the SUBTEXT strings within the TEXT
WIDTH = *scalars* or *variates*	Right-most character(s) to search in the lines of each TEXT; default * searches up to the end of each line
SKIP = *scalars* or *variates*	Number of characters to skip at the left-hand side of the lines of each TEXT; default 0

Description

The TXPOSITION directive allows you to search for strings of characters within the lines of a Genstat text structure. The text to search is specified by the TEXT parameter, and the SUBTEXT parameter specifies the strings that are to be found. You can set SUBTEXT to a single string (or to a text with just one line), if you want to search for the same string of characters within every line of the TEXT. You can set SUBTEXT to a text with as many lines as TEXT, if you want to search for different characters in each line of the TEXT. Finally, you can set TEXT to a single string, and SUBTEXT to a text with several lines, if you want to search the same string to see which of several strings might occur there. The POSITION parameter can save a variate storing the position of the first character of the SUBTEXT string(s) in each of the TEXT lines, or zero if the string has not been found.

TXPOSITION usually takes account of the case of letters (small or capital) in the strings when comparing SUBTEXT with TEXT. So for example 'GenStat' would not match with 'Genstat'. However, you can set option CASE=ignored to ignore differences in case. It will usually also treat multiple spaces as significant, but you can set option MULTISPACE=ignored to treat them all like a single space. By default, the search is from left to right (i.e. from the start to the end of each line of TEXT), but you can set option REVERSE=yes to search from right to left.

The SKIP parameter allows you to skip characters at the start of the lines of TEXT. You can supply a scalar to skip the same number of characters in every line, or a variate if you want to make different skips in each line. (So, once you have found a SUBTEXT string, you can set SKIP to its position and check whether it occurs again.) Similarly the WIDTH parameter specifies the right-most character(s) of the TEXT lines to search.

Option DISTINCT is useful if you are looking for distinct words or phrases. The left setting requires each SUBTEXT string to begin either at the start of the relevant line of TEXT, or to be preceded in that line by a separator (such as a space or comma). Similarly, the right setting

requires the SUBTEXT to end within the line of TEXT with a separator (or to be at the end of the line). The separators are specified by the SEPARATOR option.

Options: CASE, REVERSE, MULTISPACES, DISTINCT, SEPARATOR.
Parameters: TEXT, SUBTEXT, POSITION, WIDTH, SKIP.

Action with RESTRICT
TXPOSITION takes account of restrictions on any of the TEXT or SUBTEXT texts, and will search only the lines that are not excluded by the restriction. The values of the POSITION variate in the restricted units are left unchanged.

See also
Directives: TEXT, CONCATENATE, EDIT, GETLOCATIONS, TXBREAK, TXCONSTRUCT, TXFIND, TXREPLACE.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

TXREPLACE

Replaces a subtext within a text structure.

Options

NTIMES = *scalar*	Number of times to search for the OLDSUBTEXT and replace it; default 1
CASE = *string token*	Whether to treat the case of letters (small or capital) as significant when searching for the OLDSUBTEXT within the OLDTEXT (significant, ignored); default sign
MULTISPACES = *string token*	Whether to treat differences between multiple spaces and single spaces as significant when locating the OLDSUBTEXT within the OLDTEXT, or to treat them all like a single space (significant, ignored); default sign
DISTINCT = *string tokens*	Whether to require the OLDSUBTEXT to have one or more separators to its left or right within the OLDTEXT (left, right); default *
SEPARATOR = *string*	Characters to use as separators; default ' ,;:.'
SAMELINE = *string token*	Whether to ignore matches in the OLDTEXT where the OLDSUBTEXT is not all on the same line (yes, no); default no

Parameters

OLDTEXT = *texts*	Texts to be edited
NEWTEXT = *texts*	Texts with OLDSUBTEXT replaced by NEWSUBTEXT; if no NEWTEXT is supplied, the new values replace those in the corresponding OLDTEXT
OLDSUBTEXT = *texts*	Text to look for in each OLDTEXT
NEWSUBTEXT = *texts*	Text to replace OLDSUBTEXT
COLUMN = *scalars*	Position of the column within OLDTEXT where the first character of NEWSUBTEXT has been placed
LINE = *scalars*	Number of the line within OLDTEXT where the first character of NEWSUBTEXT has been placed
ICOLUMN = *scalars*	Column within OLDTEXT at which to start the search
ILINE = *scalars*	Line within OLDTEXT at which to start the search
ENDCOLUMN = *scalars*	Position of the column within OLDTEXT where the last character of NEWSUBTEXT has been placed
ENDLINE = *scalars*	Number of the line within OLDTEXT where the last character of NEWSUBTEXT has been placed
NREPLACED = *scalars*	Number of subtexts replaced

Description

The TXREPLACE directive replaces a subtext within a Genstat text structure. The text containing the subtext is specified by the OLDTEXT parameter. The OLDSUBTEXT parameter specifies the subtext to be replaced, and the NEWSUBTEXT parameter specifies the subtext to replace it. By default a single occurrence of the subtext is replaced, but you can use the NTIMES option to replace several. It you set NTIMES to a negative value, all occurrences are replaced. The NREPLACED parameter can save the number of replacements that were actually made (which may be less than NTIMES if fewer were found in the OLDTEXT). The new text (after the replacements) can be saved using the NEWTEXT parameter; if this is not set, the values of the OLDTEXT are replaced by the new text.

By default, the search treats the OLDTEXT and OLDSUBTEXT as if they were paragraphs of characters: that is, it takes no account of the line breaks within the two text structures, regarding each one as equivalent to a space. However, you can set option SAMELINE=yes to treat line breaks differently from spaces. Matches are then recognised only if they are all on a single line.

TXREPLACE usually takes account of the case of letters (small or capital) when looking for the OLDSUBTEXT within the OLDTEXT. So for example 'GenStat' would not match with 'Genstat'. However, you can set option CASE=ignored to ignore differences in case. It will usually also treat multiple spaces as significant, but you can set option MULTISPACE=ignored to treat them all like a single space.

Option DISTINCT is useful if you are looking for distinct words or phrases. The left setting requires the OLDSUBTEXT to begin either at the start of the OLDTEXT, or to be preceded in the OLDTEXT by a separator (such as a space or comma). Similarly, the right setting requires the OLDSUBTEXT to end within the OLDTEXT with a separator (or to be at the end of the OLDTEXT). The separators are specified by the SEPARATOR option.

The ICOLUMN and ILINE parameters can specify a starting column and line for the search. So you can leave an initial section of the OLDTEXT unchanged.

You can use the COLUMN parameter to save the column within the OLDTEXT where the first character of the NEWSUBTEXT has been inserted, and the LINE parameter to save its line within the OLDTEXT. These are both set to zero if the OLDSUBTEXT was not found. If NTIMES is greater than one, they save the location of the final replacement. Similarly the ENDCOLUMN and ENDLINE parameters can save the position of the last character of the NEWSUBTEXT within the OLDTEXT.

Options: NTIMES, CASE, MULTISPACES, DISTINCT, SEPARATOR, SAMELINE.
Parameters: OLDTEXT, NEWTEXT, OLDSUBTEXT, NEWSUBTEXT, COLUMN, LINE, ICOLUMN, ILINE, ENDCOLUMN, ENDLINE, NREPLACED.

Action with RESTRICT
Any restrictions are ignored.

See also
Directives: TEXT, CONCATENATE, EDIT, EQUATE, TXBREAK, TXCONSTRUCT, TXFIND, TXPOSITION.
Functions: CHARACTERS, GETFIRST, GETLAST, GETPOSITION, POSITION.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

TX2VARIATE

Converts text structures to variates.

Options

PRINT = *string token*	Controls printed output (conversions) ; default * (i.e. none)
NONNUMERIC = *string token*	How to treat non-numeric values (bestmatch, missing) default miss
YEAR = *scalar*	Year to use when calculating the day within year for the date formats that specify only months and days; default is to assume that this is any year that is not a leap year
REDEFINE = *string token*	Whether to allow a structure in the VARIATE list that has already been declared (e.g. as a text) to be redefined (yes, no); default no

Parameters

TEXT = *texts*	Text structures to convert
VARIATE = *variates*	Variate for each text, containing the numbers in each of its lines
DREPRESENTATION = *scalars*	Format to use for dates and times (stored in numerical structures)
MISSING = *texts*	Strings used to represent missing values in each text; default '*'
STATUS = *variates*	Code to indicate whether the number in each unit was read successfully (1), or with conversions (2), or unsuccessfully (0)

Description

The TX2VARIATE directive converts texts to vaiates. The texts are specified by the TEXT parameter, and are assumed to contain a single number in each of their strings. The variates are specified by the VARIATE parameter.

The DREPRESENTATION parameter specifies the format that has been used for texts that contain dates. For details, see the PRINT directive. With the formats that specify only months and days, TX2VARIATE gives the number of the day within the year. However, it needs to know whether or not the year is a leap year. You can use the YEAR option to supply the year. If this is not set, TX2VARIATE assumes that it is not a leap year.

The MISSING parameter specifies a text for each text and variate, containing the string or strings that should be treated as missing values in the conversion; by default this is the string containing a single asterisk. Blank and null lines are always treated as missing.

By default, any non-numeric strings generate a missing value in the variate. However, you can set option NONNUMERIC=bestmatch to ignore commas, and to allow for the common typing errors that the letters i or l may have been typed instead of i, or that the letters o or O may have been typed instead of 0. You can set option PRINT=conversions to print a list of the values that have been converted. Also, the STATUS parameter can save a variate with a code for each number to show whether it was read successfully with no conversions (1), or only with conversions (2), or whether it could not be read successfully (0).

If you set option REDEFINE=yes, any data structure specified by the VARIATE parameter that is not a variate will be redefined (to be a variate). Also, VARIATE then takes the setting of the TEXT parameter as its default, i.e. it will redefine that text to be a variate.

Options: PRINT, NONNUMERIC, YEAR, REDEFINE.

Parameters: TEXT, VARIATE, MISSING, STATUS.

Action with RESTRICT
TX2VARIATE takes account of restrictions on each TEXT or VARIATE. The values of the VARIATE in the units excluded by the restriction are left unchanged.

See also
Directives: TEXT, READ.
Genstat Reference Manual 1 Summary section on: Calculations and manipulation.

UNITS

Defines an auxiliary vector of labels and/or the length of any vector whose length is not defined when a statement needing it is executed.

Option

NVALUES = *scalar*	Default length for vectors

Parameter

variate or *text*	Vector of labels

Description

The UNITS directive can be used to define a default length which will then be used, if necessary, for any new vectors encountered later in the job. For example, in the statements

```
UNITS [NVALUES=20]
TEXT Subject
VARIATE [VALUES=0,1,2,4,8] Dlev
FACTOR [LEVELS=Dlev] Drug
VARIATE Age,Response; DECIMALS=0,2
```

the text Subject, the factor Drug, and the variates Age and Response are all defined to be of length 20. However, the length of the variate Dlev does not need to be set by default, but is deduced to be five from the number of values that have been specified by the VALUES option.

The READ directive will use UNITS if values are to be read into a previously undeclared vector, as will the RESTRICT directive if you use it to restrict a structure that has not yet been declared. The UNITS setting is also used by the CALCULATE directive with the EXPAND and URAND functions if their secondary argument is not specified.

The parameter of the UNITS directive allows you to specify the units structure, which is a variate or a text whose values will then be used as labels for output from regression or time-series directives, provided the vectors in the analysis have the same length as the units structure and provided also that these vectors do not have labels associated with them already.

The length of the units structure must match the value set by the NVALUES option if both are set. However, either one can be used to define the other. Thus, either

```
TEXT [VALUES=Sun,Mon,Tue,Wed,Thur,Fri,Sat] Day
UNITS Day
```

or

```
TEXT Day
UNITS [NVALUES=7] Day
```

would specify the default length for vectors to be seven. In the second example this default would be applied to Day too but, of course, its (seven) values would need to be read or defined in some other way before it could be used for labelling. If the type of the units structure has not been declared, UNITS will define it as a variate.

You can cancel the effect of a UNITS statement by

```
UNITS [NVALUES=*]
```

This means that statements that require a units structure will fail, which is the situation at the start of each job in a program. Similarly, the statement

```
UNITS *
```

cancels any reference to a units structure, but retains the default length if that has already been defined.

Option: NVALUES.
Parameter: unnamed.

See also
Directives: FACTOR, TEXT, VARIATE.
Genstat Reference Manual 1 Summary section on: Data structures.

VARIATE

Declares one or more variate data structures.

Options

NVALUES = *scalar* or *vector*	Number of units, or vector of labels; default * takes the setting from the preceding UNITS statement, if any
VALUES = *numbers*	Values for all the variates; default *
MODIFY = *string token*	Whether to modify (instead of redefining) existing structures (yes, no); default no
IPRINT = *string tokens*	Information to be used by default to identify the variates in output (identifier, extra); if this is not set, they will be identified in the standard way for each type of output

Parameters

IDENTIFIER = *identifiers*	Identifiers of the variates
VALUES = *identifiers*	Values for each variate
DECIMALS = *scalars*	Number of decimal places for output
EXTRA = *texts*	Extra text associated with each identifier
MINIMUM = *scalars*	Minimum value for the contents of each structure
MAXIMUM = *scalars*	Maximum value for the contents of each structure
DREPRESENTATION = *scalars* or *texts*	Default format to use when the contents represent dates and times

Description

The variate is probably the structure that you will use most often in Genstat. You can think of this as being just a list of numbers – a vector, in mathematical language. Variates occur for example as the response and explanatory variables in regression, as covariates and y-variables in analysis of variance, and can be used to form the matrices of correlations, similarities or sums of squares and products required for multivariate analyses.

The IDENTIFIER parameter lists the variates that are to be declared. Values can be assigned by either the VALUES option or the VALUES parameter. The option defines a common value for all the variates in the declaration, while the parameter allows them each to be given a different value. If both the option and the parameter are specified, the parameter takes precedence.

The NVALUES option allows the number of values in the variates to be defined. If this is not set, the lengths of the variates are defined from the numbers that are supplied by the VALUES option or parameter. If these too are unset, Genstat takes the length specified by the preceding UNITS statement, if any.

The DECIMALS parameter allows you to define a number of decimal places to be used by default when each variate is printed. You can associate a text of extra annotation with each variate using the EXTRA parameter. The MINIMUM and MAXIMUM parameters allow you to define lower and upper limits on the values in each variate. Genstat then prints warnings if any values outside that range are allocated to the variate. The DREPRESENTATION parameter allows a scalar or a single-valued text to be specified for each variate to indicate that the variate stores dates and times, and to define a format to be used for these, by default, when they are printed; details are given in the description of the PRINT directive.

If the MODIFY option is set to yes any existing attributes and values of the variates are retained (if still appropriate); otherwise these are lost. The IPRINT option can be set to specify how the variates will be identified in output. If IPRINT is not set, they will be identified in whatever way is usual for the section of output concerned. For example, the PRINT directive

generally uses their identifiers (although this can be changed using the IPRINT option of PRINT itself), while the ANOVA directive will print the identifier and the extra text for each y-variate.

Options: NVALUES, VALUES, MODIFY, IPRINT.
Parameters: IDENTIFIER, VALUES, DECIMALS, EXTRA, MINIMUM, MAXIMUM, DREPRESENTATION.

See also
Directives: FACTOR, TEXT, UNITS.
Procedure: TX2VARIATE.
Genstat Reference Manual 1 Summary section on: Data structures.

VCOMPONENTS

Defines the variance-components model for REML.

Options

FIXED = *formula*	Fixed model terms; default *
ABSORB = *factor*	Defines the absorbing factor (appropriate only when REML option METHOD=Fisher); default * i.e. none
CONSTANT = *string token*	How to treat the constant term (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
CADJUST = *string token*	What adjustment to make to covariates before analysis (mean, none); default mean
RELATIONSHIP = *matrix*	Defines relationships constraining the values of the components; default *
SPLINE = *formula*	Defines random cubic spline terms to be generated: each term must contain only one variate, if there is more than one factor in a term, separate splines are calculated for each combination of levels of the factors
EXPERIMENTS = *factor*	Factor defining the different experiments in a multi-experiment (meta-) analysis

Parameters

RANDOM = *formula*	Random model terms and the residual variance
INITIAL = *scalars*	Initial values for each component
CONSTRAINTS = *string tokens*	How to constrain each variance component and the residual variance (none, positive, fixrelative, fixabsolute); default none

Description

The VCOMPONENTS directive specifies the linear mixed model to be fitted by subsequent REML statements. There are usually two parts of the mixed model to be defined, namely the fixed and random model terms. In addition, it is possible to specify terms to generate cubic splines.

Random effects are used to describe the effects of factors where the values present in the experiment can be considered as a random selection of values from some large homogeneous population. Inference about this population can then be made, for example estimation of its variance. Predictions of random effects may also be of interest. Fixed effects are used to describe treatments imposed in an experiment where the effect of those specific choices of treatment are of interest. Cubic spline terms can be included to investigate smooth non-linear departures from the model specified.

For example, consider a split-plot experiment used to assess the effects on yield of three oat varieties with four levels of nitrogen application. Here specific levels of nitrogen application have been used and the aim is to estimate the effects of these levels; so they would be considered as fixed effects in the model, as would the three oat varieties. However, the effects of the actual blocks and plots in the experiment are not of interest in themselves, but they do provide a means of estimating the variability of the more general population of blocks and plots in order to get an estimate of background variation against which to compare the fixed effects. Blocks and plots would therefore be defined as random effects. In this case, the fixed effects correspond to the effects used as treatments in ANOVA and the random effects would correspond to the blocking factors in ANOVA.

In general, both the fixed and random parts of the model are constructed from several factors

or variates. The structure of both parts is specified using model formulae and can contain both factors and variates with the usual adding, crossing or nesting operators.

The fixed terms in the model are defined by a model formula supplied using the `FIXED` option, and the random model terms are defined by a model formula supplied by the `RANDOM` parameter. Thus, for example, the model for the split-plot experiment described above would be specified by

```
VCOMPONENTS [FIXED=Nitrogen*Variety] \
            RANDOM=Block/Wplot/Subplot
```

where `Nitrogen` and `Variety` are factors indicating the treatments applied to each unit, and `Block`, `Wplot` and `Subplot` are factors indicating the block, wholeplot (within block) and subplot (within wholeplot) to which each unit belongs.

The default fixed model consists of just the constant term, which then becomes the grand mean. The constant term can be omitted by setting option `CONSTANT=omit`, provided a fixed model has been specified. If the random model is unset, only a single source of variation (the residual component called `*units*`) is used.

When covariates are included in the fixed or random models, by default they are automatically centred before analysis. However, you can set option `CADJUST=none` to specify that the uncentred covariates are to be used instead.

The `FACTORIAL` option is used to set a limit on the number of factors and variates allowed in each fixed term; any term containing more than that number is deleted from the model.

The `SPLINE` option can be used to generate cubic spline terms to be fitted as part of the random model. The smoothing parameter is estimated by REML and the fitted spline is interpreted as a BLUP (best linear unbiased predictor). If a term consists of a single variate, for example `SPLINE=X`, a cubic spline will be generated using all distinct covariate values present as knots, with weighting for replicate points. If factors are included, for example `SPLINE=N.V.X` where `N` and `V` are factors, separate cubic splines will be generated for each level of the combined factor `N.V`. The knot points for the splines will be generated as the set of distinct values in `X`, and the same knot points will be used at each level of `N.V`.

The `EXPERIMENTS` option is used when a combined (or *meta*-) analysis is being defined over several experiments. The setting is a factor identifying the experiment to which each unit belonged. When this option has been set, a different residual term will be set up for each level of the `EXPERIMENTS` factor. In the simplest case, this means that a different residual variance will be used for each experiment. For more complex cases, different correlated error models can be applied to separate experiments using the `VRESIDUAL` directive. In either case, the factors and variates for the separate experiments should be concatenated into structures which run over all the experiments. For example, consider an experiment set up at two sites to compare a set of 24 varieties in four replicates. In one site the experiment was laid out as a grid of eight rows by 12 columns, in the other a grid of 16 rows by six columns was used. In these circumstances, a single set of factors (of length 192) can be used to specify the design, using factors to describe variety, rows and columns, plus a factor `expt` defining the allocation of units to experiments. Note that the factor `row` will have 16 levels and `col` will have 12 levels. The restriction of site 1 to 8 rows and site 2 to 6 columns is specified using the `VRESIDUAL` directive. Where some factors differ between experiments, these should be defined on the units relevant to the appropriate experiment(s) and missing elsewhere.

For random terms, initial values for the ratio of variance components to the error variance (the gamma ratios) are supplied using the `INITIAL` parameter, and you can impose general constraints on the variance components using the `CONSTRAINTS` parameter, or equality constraints between components using the `RELATIONSHIP` option. By default, all the gamma ratios have initial values of one. The `CONSTRAINTS` parameter can request that any variance component should be held positive or fixed at its initial value. The default setting, `none`, allows the variance components to become negative, provided the overall estimated variance-covariance

matrix for the data remains positive definite. The RELATIONSHIP option can be used to define linear relationships between the variance components, for example that component A should be constrained to be twice component B.

Covariance models for random terms, including unknown parameters to be estimated, can be specified using the VSTRUCTURE directive.

The ABSORB option allows you to specify a factor from either the fixed or the random model to act as an absorbing factor for the model. Note that the absorbing factor is ignored for the AI algorithm with sparse matrix methods: that is, this option is relevant only when the METHOD option of REML is set to Fisher. The absorbing factor is used to divide the model terms into two groups; this partition is then used in calculations during the fitting process to reduce the size of the matrices that have to be inverted and stored. Use of an absorbing factor can therefore save computing time and data space. However, although exactly the same model is fitted when an absorbing factor is used, some of the standard errors are unavailable. A good choice of absorbing factor might be a factor with a large number of levels, or any factor whose effects and standard errors are not of interest.

Options: FIXED, ABSORB, CONSTANT, FACTORIAL, CADJUST, RELATIONSHIP, SPLINE, EXPERIMENTS.
Parameters: RANDOM, INITIAL, CONSTRAINTS.

Action with RESTRICT

You can restrict the set of units to be included in the analysis by restricting any of the factors or variates in the fixed and random models defined by VCOMPONENTS, or by restricting any of the y-variates in the subsequent REML statement. However, if more than one of these vectors is restricted, all must be restricted to the same set of units.

See also

Directives: REML, VSTRUCTURE, VRESIDUAL, VPEDIGREE, VSTATUS.
Procedures: FCONTRASTS, FDIALLEL, VFMODEL, VRACCUMULATE, VRMETAMODEL.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VCYCLE

Controls the operation of the REML algorithm.

Options

CONVERGENCE = *string token*	Type of criterion for assessing convergence (deviance, parameter); default * uses the deviance with the average-information algorithm, and the variance parameter values for the Fisher scoring algorithm
CRITERIONVALUE = *scalar*	Sets the convergence criterion value; default * i.e. determined automatically
STEPLENGTH = *scalar*	Sets the default relative step size for the average-information algorithm; default * i.e. determined automatically
NDENSE = *scalar*	Number of equations to use as dense in the average-information algorithm; default * uses all fixed model terms as dense
EQORDER = *string token*	Method to use to reorder the mixed model equations for fitting (none, a, b); default b

No parameters

Description

VCYCLE allows you to modify various aspects of the REML algorithm. The CONVERGENCE option specifies the type of criterion to use to assess convergence. There are two possibilities, each of which is used as the default for one of the fitting algorithms. For the average-information algorithm the default is to check for convergence in deviance, whereas the Fisher method checks the variance parameter values. The criterion value can be specified by the CRITERIONVALUE option. The defaults differ according to the type of criterion. For assessing changes in variance parameter values a multiplier of 0.005 is used. So, for convergence, the change in every variance parameter s must be less than $0.005 \times s$. When assessing change in deviance, convergence occurs when the absolute change in the deviance is less than 0.0001.

The STEPLENGTH option allows you to change the default step size for the average-information algorithm. Valid values are between zero and one, and the value is the proportion of the average-information step taken. The default is to start with small steps and work up to full steps.

The NDENSE option allows you to manipulate the number of equations used as dense in the average-information algorithm. The default includes all the fixed model terms. This option is likely to be used only by advanced users. If NDENSE is set, the value may be modified by the algorithm so that model terms are not split between the dense and sparse sections. Note that Wald tests (dropping terms) are not available for terms in the sparse section.

The EQORDER option controls the order in which the mixed model equations are solved, with settings:

none	processes the equations in the order in which they are specified in the model;
a	method A; and
b	method B (default).

This option needs to be set only rarely as method B, which corresponds to the ASReml option setting !EQORDER 3 (introduced to become the default in ASReml Release 2), is generally the best. Method A corresponds to the ASReml option setting !EQORDER 1 (which was the default in ASReml Release 1). For further details, see *ASReml User Guide Release 2*.

Options: CONVERGENCE, CRITERIONVALUE, STEPLENGTH, NDENSE, EQORDER.
Parameters: none.

See also
Directive: REML.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VDISPLAY

Displays further output from a REML analysis.

Options

PRINT = *string tokens*	What output to present (model, components, effects, means, stratumvariances, monitoring, vcovariance, deviance, Waldtests, missingvalues, covariancemodels); default mode, comp, Wald, cova
CHANNEL = *identifier*	Channel number of file, or identifier of a text to store output; default current output file
PTERMS = *formula*	Terms (fixed or random) for which effects or means are to be printed; default * implies all the fixed terms
PSE = *string token*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimates, none); default diff
CFORMAT = *string token*	Whether printed output for covariance models gives the variance matrices or the parameters (variancematrices, parameters); default vari
FMETHOD = *string token*	Controls whether and how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto

Parameter

REML save structures	Save structure containing the details of each analysis; default is to take the save structure from the latest REML analysis

Description

The VDISPLAY directive allows further output to be produced from one or more REML analyses without having to repeat all the calculations.

Information from a REML analysis can be stored using the parameter SAVE in the REML statement for use in the SAVE parameter of VDISPLAY. Several SAVE structures can be specified, corresponding to the analyses of several different variates. By default, the save structure for the last y-variate analysed is saved automatically and used by VDISPLAY.

The options of VDISPLAY are the same as those that control output from REML: PRINT, PTERMS, PSE, CFORMAT and FMETHOD, plus the CHANNEL option which allows output to be directed to another output channel or into a text structure. The available settings of PRINT are identical to those in REML. For example, the commands

```
VCOMPONENTS [FIXED=Nitrogen*Variety] RANDOM=Block/Wplot/Splot
REML [PRINT=model,wald,components] Yield
VDISPLAY [PRINT=effects]
```

print the effects for the fixed terms after the analysis, without having to re-run the algorithm.

Options: PRINT, CHANNEL, PTERMS, PSE, CFORMAT, FMETHOD.
Parameter: unnamed.

See also
Directives: REML, VCOMPONENTS, VSTRUCTURE, VRESIDUAL, VPREDICT, VKEEP, VSTATUS.
Procedures: VAIC, VCHECK, VGRAPH, VPLOT, VDFIELDRESIDUALS, VFUNCTION, VHERITABILITY, VLSD, VMCOMPARISON, VRCHECK, VTCOMPARISONS.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VKEEP

Copies information from a REML analysis into Genstat data structures.

Options

RESIDUALS = *variate*	Residuals from the analysis
FITTEDVALUES = *variate*	Fitted values from the analysis
SIGMA2 = *scalar*	Variance component for the lowest stratum
VCOVARIANCE = *symmetric matrix*	Variance-covariance matrix for the estimates of the variance components
VESTIMATES = *variate*	Saves a vector of all parameters in the variance model
VARESTIMATES = *symmetric matrix*	Variance-covariance matrix for the parameters in the variance model (as saved by VESTIMATES)
VLABELS = *text*	Vector of text labels for the VESTIMATES and VARESTIMATES structures
MVESTIMATES = *variate*	Estimates of missing values
MVSE = *variate*	Standard errors of missing-value estimates
MVUNITS = *variate*	Unit numbers of missing values
ALLEFFECTS = *variate*	Full set of estimated fixed and random effects
ALLVCOVARIANCE = *symmetric matrix*	Variance-covariance matrix for the full set of fixed and random effects not associated with the absorbing factor
DEVIANCE = *scalar*	Residual deviance from fitting the full fixed model
DF = *scalar*	Residual degrees of freedom after fitting the full fixed model
SUBDEVIANCE = *scalar*	Residual deviance after fitting the submodel of the fixed model
SUBDF = *scalar*	Residual degrees of freedom after fitting the submodel of the fixed model
RSS = *scalar*	Residual sum of squares from fitting the FIXED model by general least squares with a covariance matrix derived from the estimated variance components
INDEX = *variate*	Index of units included in the analysis
MODELS = *pointer*	Pointer to formulae giving the fixed, random, spline and residual terms fitted
RMATRIX = *pointer*	Saves details of the covariance model fitted to the residual
RMETHOD = *string token*	Which random terms to use when calculating RESIDUALS (final, all, notspline); default uses the setting from the REML statement
CFORMAT = *string token*	Whether the covariance matrices or the parameters are saved for a COVARIANCEMODEL (variancematrices, parameters); default vari
UVCOVARIANCE = *symmetric matrix*	Unit-by-unit variance-covariance matrix
DFFIXED = *scalar*	Number of degrees of freedom in the fixed model
DFRANDOM = *scalar*	Number of degrees of freedom in the random model
FMETHOD = *string token*	Controls how to calculate F-statistics for fixed terms (automatic, none, algebraic, numerical); default auto
WMETHOD = *string token*	Controls which Wald statistics are saved (add, drop);

	default `drop`
`WORKSPACE` = *scalar*	Saves the workspace setting that was used in the `REML` command
`YVARIATE` = *dummy*	Dummy to be set to the y-variate of the analysis
`EXIT` = *scalar*	Exit status of the fit (0 if successful)
`SAVE` = *REML save structure*	Save structure from the required analysis; default `*` takes the save structure from the latest `REML` statement

Parameters

`TERMS` = *formula*	Terms for which information is to be saved
`COMPONENTS` = *scalars*	Estimated variance components
`COVARIANCEMODEL` = *pointers*	Saves details of the covariance model fitted to a random term
`MEANS` = *tables*	Table of predicted means for each term
`SEDMEANS` = *symmetric matrices*	Standard errors of differences between the predicted means
`VARMEANS` = *symmetric matrices*	Variance-covariance matrix of the means
`EFFECTS` = *tables*	Table of estimated regression coefficients for each term
`SEDEFFECTS` = *symmetric matrices*	Standard errors of differences between the estimated parameters of each term
`VAREFFECTS` = *symmetric matrices*	Variance-covariance matrix of the effects of a term
`DESIGNMATRIX` = *matrices*	Saves the design matrix for the term
`SPLBLUP` = *pointers*	Best linear unbiased predictors for spline terms, saved in a pointer with a variate for each combination of the levels of the factors in the term
`SPLDESIGN` = *pointers*	Design matrices (Z) for spline terms, saved in a pointer with a matrix for each combination of the levels of the factors in the term
`SPLX` = *pointers*	Knot points for spline terms, saved in a pointer with a variate for each combination of the levels of the factors in the term
`SPLSMOOTH` = *pointers*	Smoothing parameters estimated for spline terms, saved in a pointer with a scalar for each combination of the levels of the factors in the term
`CADJUSTMENT` = *scalars*	For a term involving covariates, saves the adjustment made to its values during the analysis
`WALD` = *scalars*	Wald statistic (fixed terms only)
`FSTATISTIC` = *scalars*	F statistics (fixed terms only)
`NDF` = *scalars*	Numerator d.f. (fixed terms only)
`DDF` = *scalars*	Denominator d.f. (fixed terms only)

Description

The `VKEEP` directive is used to copy results from a `REML` analysis into Genstat data structures. Genstat automatically stores the save structure for the last y-variate that was analysed using `REML`, and by default this save structure provides the information for `VKEEP`. Alternatively, you can save the information from a `REML` analysis in a save structure using the `SAVE` parameter in the `REML` directive, then access the information by specifying the same structure in the `SAVE` option of `VKEEP`.

Overall information from the analysis is saved using the options of `VKEEP`, while the parameters are used to save information for specific model terms. The terms (fixed, random or a mixture) for which you require information are defined by a formula using the `TERMS`

parameter. The other parameters can then be used to specify structures for saving information for each of the model terms.

Options RESIDUALS and FITTEDVALUES are used to specify variates to hold the residuals and fitted values, which are defined according to the setting of the RMETHOD option, as for the REML directive. The residual variance can be stored in a scalar using option SIGMA2.

The variance-covariance matrix for the estimates of the variance component can be saved using the VCOVARIANCE option. (The estimates themselves are saved using the COMPONENTS parameter, as described below.)

The VESTIMATES option is used to save a variate containing all the variance parameters estimated in the model. The VARESTIMATES option can supply a symmetric matrix to save the variance-covariance matrix for the estimates of the variance parameters, matching the ordering and contents of VESTIMATES. The vector of labels for these parameters can be saved the VLABELS option. The ALLEFFECTS option allows you to save the full set of fixed and random effects, excluding those in the absorbing factor model, and the ALLVCOVARIANCE option can be used to store their variance-covariance matrix. This matrix will often be very large, and is useful only for looking at covariances between effects associated with different model terms, since the variance-covariance matrices for individual model terms can be stored using the VAREFFECTS parameter. The unit-by-unit variance-covariance matrix can be saved using the UVCOVARIANCE option (and this may be even larger). This uses the random and residual terms, but not spline terms. It cannot be formed if the model contains sparse inverse covariance matrices, for example from VPEDIGREE.

The MVESTIMATES option can save a variate containing estimates of the missing values, the MVSE option saves their standard errors, and the MVUNITS option saves a list of the units that are missing.

The residual deviance from fitting the full fixed model or the submodel can be saved using options DEVIANCE and SUBDEVIANCE respectively, and the associated residual degrees of freedom can be saved using options DF and SUBDF. The degrees of freedom fitted by the (full) fixed model can be saved by the DFFIXED option, and the degrees of freedom in the random model can be saved by the DFRANDOM option. The RSS option can save the residual sum of squares from fitting the fixed model by generalized least squares.

The INDEX option saves an index of the units that were included in the the analysis. (This will depend on the patterns of missing values, if any, and the setting of the MVINCLUDE option of REML.)

The MODELS option can be used to save a pointer, with labels 'Fixed', 'Spline', 'Random' and 'Residual', containing formulae for the model terms fitted as fixed, spline, random or residual terms. The labels can be specified in either lower or upper case, or any mixture. The YVARIATE option can be set to a dummy to point to the variate that was analysed (i.e. the variate defined by the Y parameter of REML).

The formula specified by the TERMS parameter is expanded to give a series of model terms. The other parameters of VKEEP are taken in parallel with these terms. The string 'Constant' can be used within the formula to save structures associated with the constant term.

The COMPONENTS parameter allows you to save the estimated variance component for each random term in the TERMS list. Details of the covariance model fitted to each random term can be saved using the COVARIANCEMODEL parameter. The information is saved in a pointer. The contents of the pointer depend upon the complexity of the covariance model fitted and the setting of the CFORMAT parameter. First we consider the default setting: CFORMAT=variancematrices. If no covariance model has been fitted, the pointer will have two elements for the scalar (variance component) and the covariance matrix (identity – a diagonal matrix with number of rows equal to the number of levels of the term). If a covariance model has been fitted, the component matrices used to construct the model will be saved. The full covariance matrix can then be generated by taking a direct product of the component

matrices and multiplying by the scalar. Alternatively, if CFORMAT=parameters, the pointer contains the component parameters of the model. The RMATRIX option provides an alternative way of saving the covariance model fitted to the residual term.

Tables of means for each term can be saved using the MEANS parameter, and standard errors of differences between the means are saved by SEDMEANS. You can also save the estimated variance-covariance matrix for the means of each term using parameter VARMEANS.

For example, you can save table of means and variance-covariance matrices of the means for terms A and B, by the command

```
VKEEP A+B; MEANS=MeanA,MeanB; VARMEANS=VarmeanA,VarmeanB
```

MeanA and MeanB will then be tables containing predicted means for factors A and B, and VarmeanA and VarmeanB will be symmetric matrices containing the variances and covariances between the table cells.

The EFFECTS parameter is used to save tables of estimated parameters. A symmetric matrix of the standard errors of differences between the effects of each term can be saved using parameter SEDEFFECTS, and the estimated variance-covariance matrix for the parameters can be saved using parameter VAREFFECTS. The DESIGNMATRIX parameter saves the design matrix used to fit the effects of each term.

You can save details of splines that have been fitted for each term using the SPLBLUP, SPLDESIGN, SPLX and SPLSMOOTH parameters. The information is saved in pointers with an element for each combination of the levels of the factors in the term (i.e. for each spline that has been fitted). The pointers elements are variates for SPLBLUP (best linear unbiased predictors) and SPLX (knot points), matrices for SPLDESIGN (design matrices), and scalars for SPLSMOOTH (smoothing parameters).

If the term involves a covariate, the CADJUSTMENT parameter can save the adjustment that will have been made to its values during the analysis. This will be zero if option CADJUST was set to none when the fixed and random models were defined by VCOMPONENTS. Alternatively, if CADJUST had its default setting of mean, each covariate will have been centred by subtracting its (weighted) mean.

The Wald statistic for a fixed term can be saved using the WALD parameter. The WMETHOD option controls whether these are from the table where terms are added sequentially to the model, or that where terms are dropped from the full fixed model. The associated F statistic, and its numerator and denominator numbers of degrees of freedom, can be saved by the FSTATISTIC, NDF and DDF parameters, respectively. The FMETHOD option specifies which algorithm to use to calculate the denominator numbers of degrees of freedom. The default, automatic, will use any stored values that have been calculated for this analysis by earlier REML, VDISPLAY or VKEEP statements; otherwise it will choose automatically between the two available methods. (See REML for more details.)

The WORKSPACE option can save the workspace setting that was used in the REML command that performed the analysis, and the EXIT option can save a code defining the exit status of the analysis. The codes (which are also used in the EXIT parameter of REML) are as follows:

0 analysis was completed successfully;
1 analysis did not converge within the specified number of iterations (but no fault occurred);
2 the fit was halted because no progress could be made;
3 the fit was halted the log-likelihood was diverging;
4 a parameter has gone out of bounds;
5 insufficient workspace;
6 no save structure is available (no REML command or a fault occurred (may be set by VKEEP but not by REML);
7 value of deviance at final iteration larger than at previous iteration(s);
−1 the algorithm performed an iteration but failed for an indeterminate reason before the exit status was established;

−2 a failure occurred prior to calling the fitting algorithm.

Options: RESIDUALS, FITTEDVALUES, SIGMA2, VCOVARIANCE, VESTIMATES, VARESTIMATES, VLABELS, MVESTIMATES, MVUNITS, ALLEFFECTS, ALLVCOVARIANCE, DEVIANCE, DF, SUBDEVIANCE, SUBDF, RSS, INDEX, MODELS, RMATRIX, RMETHOD, CFORMAT, UVCOVARIANCE, DFFIXED, DFRANDOM, FMETHOD, WMETHOD, WORKSPACE, YVARIATE, EXIT, SAVE.

Parameters: TERMS, COMPONENTS, COVARIANCEMODEL, MEANS, SEDMEANS, VARMEANS, EFFECTS, SEDEFFECTS, VAREFFECTS, DESIGNMATRIX, SPLBLUP, SPLDESIGN, SPLX, SPLSMOOTH, CADJUSTMENT, WALD, FSTATISTIC, NDF, DDF.

See also

Directives: REML, VCOMPONENTS, VSTRUCTURE, VDISPLAY, VPREDICT, VSTATUS.

Procedures: VAIC, VCHECK, VFRESIDUALS, VFIXEDTESTS, VFUNCTION, VHERITABILITY, VLSD, VMCOMPARISON, VSPREADSHEET, VUVCOVARIANCE.

Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VPEDIGREE

Generates an inverse relationship matrix for use when fitting animal or plant breeding models by REML.

Options

SEX = *string token*	Possible sex categories of parents (fixed, either); default fixe
UNKNOWN = *scalar*	Value to be treated as unknown

Parameters

INDIVIDUALS = *factors*	Individuals on which data has been measured
MALEPARENTS = *factors*	Male parents of the progeny
FEMALEPARENTS = *factors*	Female parents of the progeny
INVERSE = *pointer*	Inverse relationship matrix in sparse matrix form
POPULATION = *variates*	Full list of identifiers generated from the individuals and parents

Description

VPEDIGREE is used to generate a sparse inverse relationship matrix for use when fitting animal (or plant) breeding models by REML. The algorithm requires three parallel factors as input. The numerical levels of these factors must give identifiers for the individuals from which data are available (INDIVIDUALS) and the identifiers for the male and female parents for each individual (MALEPARENTS and FEMALEPARENTS). Note that an individual may appear as both progeny and a parent (for example, when data has been taken from several generations) and conversely, that if an identifier appears in more than one list then it is assumed to refer to a single individual. Also, the algorithm does not take account of labels, so where textual labels are used the labels vectors of the three factors should be identical in order to generate matching levels vectors and thus avoid errors. A complete list of all individuals in the three factors is compiled and can be saved using the POPULATION option, and on output, the three factors will be redefined with this list as their levels vector.

The inverse relationship matrix that is generated is held in a special sparse matrix form (that is, only non-zero values are stored), using a pointer. This is usable in the VSTRUCTURE directive but not, currently, elsewhere in Genstat. The second element of the pointer is a variate storing the non-zero values of the inverse matrix in lower-triangular order. The first element of the pointer is an integer index vector. This vector is not a standard Genstat data structure, and so cannot be used except by VSTRUCTURE.

By default, it is assumed that an individual can act as either a male or female parent but not both. Option SEX=either can be used to specify that individuals can act as both male and female parents. This may be useful, for example, in plant breeding analyses.

Missing values in any of the factors will be treated as coding for unknown individuals. Option UNKNOWN allows you to specify an additional scalar value used to represent unknown individuals.

Options: SEX, UNKNOWN.
Parameters: INDIVIDUALS, MALEPARENTS, FEMALEPARENTS, INVERSE, POPULATION.

Action with RESTRICT

VPEDIGREE ignores any restrictions on the factors.

See also
Directives: REML, VCOMPONENTS, VSTRUCTURE, VRESIDUAL, VSTATUS.
Procedure: VFPEDIGREE.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VPREDICT

Forms predictions from a REML model.

Options

PRINT = *string tokens*	What to print (description, predictions, se, sed, avesed, vcovariance); default desc, pred, se, aves
CHANNEL = *scalar*	Channel number for output; default * i.e. current output channel
MODEL = *formula*	Indicates which model terms (fixed and/or random) are to be used in forming the predictions; default * includes all the fixed terms and relevant random terms
OMITTERMS = *formula*	Specifies terms to be excluded from the MODEL; default * i.e. none
FACTORIAL = *scalar*	Limit on the number of factors or variates in each term in the models specified by MODEL or OMITTERMS; default 3
PRESENTCOMBINATIONS = *identifiers*	Lists factors for which averages should be taken across combinations that are present
WEIGHTS = *tables*	One-way tables of weights classified by factors in the model; default *
PREDICTIONS = *table* or *scalar*	To save the predictions; default *
SE = *table* or *scalar*	To save standard errors of predictions; default *
SED = *symmetric matrix*	To save standard errors of differences between predictions; default *
VCOVARIANCE = *symmetric matrix*	To save variances and covariances of predictions; default *
SAVE = *REML save structure*	Specifies the save structure from which to predict; default * i.e. that from most recent REML

Parameters

CLASSIFY = *vectors*	Variates and/or factors to classify table of predictions
LEVELS = *variates*, *scalars* or *texts*	To specify values of variates and/or levels of factors for which predictions are calculated
PARALLEL = *identifiers*	For each vector in the CLASSIFY list, allows you to specify another vector in the CLASSIFY list with which the values of this vector should change in parallel (you then obtain just one dimension in the table of predictions for these vectors)
NEWFACTOR = *identifiers*	Identifiers for new factors that are defined when LEVELS are specified

Description

The VPREDICT directive can be used after the REML directive to produce predictions of the values of the response variate at particular values of the variables in the fixed or random models. By default the predictions are from the most recent REML analysis, but you can use another analysis by supplying its save structure using the SAVE option.

The CLASSIFY parameter specifies those variates or factors to be included in the table of predictions, and the LEVELS parameter supplies the values at which the predictions are to be made. For a factor, you can select some or all of the levels, while for a variate you can specify any set of values. A single level or value is represented by a scalar; several levels or values must

be combined into a variate (which may of course be unnamed). Alternatively, if the factor has labels, you can use these to select the levels for prediction by setting LEVELS to a text. A missing value in the LEVELS parameter is taken to stand for all the levels of a factor, or the mean value of a variate.

The PARALLEL parameter allows you to indicate that a factor or variate should change in parallel with another factor or variate. Both of these should have the same number of values specified for it by the LEVELS parameter of VPREDICT. The predictions are then formed for each set of corresponding values rather than for every combination of these values. For example, suppose we had fitted a fixed model containing a factor Treatment, a variate Time representing the times when measurements were made, and a variate Timesqrd containing the squares of the times. We could then put

```
VPREDICT Treatment,Timesqrd,Time; PARALLEL=*,Time,*;\
         LEVELS=*,!(0,1,9,25,49,81),!(0,1,3,5,7,9)
```

to produce predictions at times 0, 1, 3, 5, 7 and 9 for the treatments. The PARALLEL parameter specifies that Timesqrd should change in parallel to Time, so that we obtain predictions only for matching values of Time and Timesqrd.

When you specify LEVELS, VPREDICT needs to define a new factor to classify that dimension of the table. By default this will be an unnamed factor, but you can use the NEWFACTOR parameter to give it an identifier. The EXTRA attribute of the factor is set to the name of the corresponding factor or variate in the CLASSIFY list; this will then be used to label that dimension of the table of predictions.

The prediction calculations consist of two steps. The first step is to calculate a table of fitted values. The MODEL, OMITTERMS and FACTORIAL options specify the model to use for this. The formula specified by MODEL is expanded into a list of model terms, deleting any that contain more variates of factors than the limit specified by the FACTORIAL option. Then, any terms in the formula specified by OMITTERMS are removed.

The second step averages the fitted values over the classifications that are not in the list that was supplied by the CLASSIFY parameter. The WEIGHTS option can supply one-way tables classified by any of the factors in the model. These are used to calculate the weight to be used for each fitted value when calculating the averages. Equal weights are assumed for any factor for which no table of weights has been supplied. (Note, this differs from the default in PREDICT, which uses *marginal weights*; see the PREDICT option ADJUSTMENT for details.) In the averaging all the fitted values are generally used. However, if you define a list of factors using the PRESENTCOMBINATIONS option, any combination of levels of these factors that does not occur in the data will be omitted from the averaging. Where a prediction is found to be inestimable, i.e. not invariant to the model parameterization, a missing value is given.

Printed output is controlled by settings of the PRINT option with settings:

description	describes the terms and standardization policies used when forming the predictions,
predictions	prints the predictions,
se	produces predictions and standard errors,
sed	prints standard errors for differences between the predictions,
avesed	prints the average standard error of difference of the predictions, and
vcovariance	prints the variance and covariances of the predictions.

By default descriptions, predictions, standard errors and an average standard error of differences are printed. You can also save the results, using the PREDICTIONS, SE, SED and VCOVARIANCE options. You can send the output to another channel, or to a text structure, by setting the CHANNEL option.

Options: PRINT, CHANNEL, MODEL, OMITTERMS, FACTORIAL, PRESENTCOMBINATIONS, WEIGHTS, PREDICTIONS, SE, SED, VCOVARIANCE, SAVE.
Parameters: CLASSIFY, LEVELS, PARALLEL.

See also
Directives: REML, VCOMPONENTS, VDISPLAY, VKEEP, PREDICT.
Procedures: VFUNCTION, VLSD, VMCOMPARISON, HGPREDICT.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VRESIDUAL

Defines the residual term for a REML analysis, or the residual term for an experiment within a meta-analysis (combined analysis of several experiments).

Options

EXPERIMENT = *scalar*	Level of the EXPERIMENTS factor for which the residual is being defined
TERM = *formula*	Model term to be used as the residual
FORMATION = *string token*	Whether the structure is formed by direct product construction or by definition of the whole matrix (direct, whole); default dire
VARIANCE = *scalar*	Allows an initial estimate to be provided for the residual variance of the experiment
CONSTRAINT = *string token*	Allows the residual variance to be fixed at its initial value (fix, positive) default posi
COORDINATES = *matrix* or *variates*	Coordinates of the data points to be used in calculating distance-based models

Parameters

MODELTYPE = *string tokens*	Type of covariance model associated with the term(s), or with individual factors in the term(s) if FORMATION=direct (identity, fixed, AR, MA, ARMA, power, boundedlinear, circular, spherical, linearvariance, banded, correlation, antedependence, unstructured, diagonal, uniform, FA, FAequal) default iden
ORDER = *scalar*	Order of model
HETEROGENEITY = *string token*	Heterogeneity for correlation matrices (none, outside); default none
METRIC = *string token*	How to calculate distances when MODEL=power (cityblock, squared, euclidean); default city
FACTOR = *factors*	Factors over which to form direct products
MATRIX = *identifiers*	To define matrix values for the term or the factors when MODEL=fixed
INVERSE = *identifiers*	To define values for matrix inverses (instead of the fixed matrices themselves) when MODEL=fixed
INITIAL = *identifiers*	Initial parameter values for each correlation matrix
CONSTRAINTS = *texts*	Texts containing strings none, fix or positive to define constraints for the parameters in each model
EQUALITYCONSTRAINTS = *variates*	Non-zero values in the variate indicate groups of parameters whose values are to be constrained to be equal

Description

VRESIDUAL is used to define the residual term for a REML analysis or to define separate residual terms for different experiments within a multi-experiment (or meta-) analysis. The TERM option is used to specify the formula for the residual term. This term need not have been specified previously by the VCOMPONENTS statement.

For a single experiment, VRESIDUAL can be used to impose a covariance structure on the residual term. This could also be done by specifying the covariance structure using

VSTRUCTURE, but VRESIDUAL has the advantage that the algorithm then checks that the correct residual term is used.

In a multi-site experiment, VRESIDUAL can be used to specify a different residual model for each separate experiment. The EXPERIMENT option is used to define the experiment(s) for which the model is to be used.

The VARIANCE option is used to give an initial value for the residual variance in the current experiment(s). You can set option CONSTRAINT=fix to fix the residual variance at the initial value rather than estimating it (as a positive value).

The definition of the residual terms then follows mainly as for the definition of correlated error terms through VSTRUCTURE The exception is that power models can be defined only in terms of the coordinates of the data points, not by specifying coordinates for the factor levels. (So the DISTANCES and COORDINATES parameters of VSTRUCTURE are not present in VRESIDUAL.)

For a multi-experiment analysis, the factors and variates for the separate experiments should be concatenated into structures which run over all the experiments. For example, consider an experiment set up at two sites to compare a set of 24 varieties in four replicates. In one site the experiment was laid out as a grid of eight rows by 12 columns, in the other a grid of 16 rows by six columns was used. In these circumstances, a single set of factors (of length 192) can be used to specify the design, using factors to describe variety, rows and columns, plus a factor expt defining the allocation of units to experiments.

```
VCOMPONENTS [FIXED=Variety; EXPERIMENTS=Expt]
VRESIDUAL [EXPERIMENT=1; TERM=Row.Col] MODEL=AR,AR; \
  ORDER=1,1; FACTOR=Row,Col
VRESIDUAL [EXPERIMENT=2; TERM=Row.Col] MODEL=AR,AR; \
  ORDER=1,1; FACTOR=Row,Col
```

Where some factors differ between experiments, these should be defined on the units relevant to the appropriate experiment(s) and missing elsewhere. When an EXPERIMENTS factor has been defined, the default action of the MVINCLUDE option of REML is changed to include units with missing y-values and missing factor levels.

Options: EXPERIMENT, TERM, FORMATION, VARIANCE, CONSTRAINT, COORDINATES
Parameters: MODEL, ORDER, HETEROGENEITY, METRIC, FACTOR, MATRIX, INVERSE, INITIAL, CONSTRAINTS, EQUALITYCONSTRAINTS.

See also
Directives: REML, VCOMPONENTS, VSTRUCTURE, VPEDIGREE, VSTATUS.
Procedures: VAMETA, VRMETAMODEL.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VSTATUS

Prints the current model settings for REML.

Option

PRINT = *string tokens*	What to print (model); default mode

No parameters

Description

VSTATUS can be used to print out and hence check the fixed and random models and covariance structures as set up by the VCOMPONENTS and VSTRUCTURE directives, prior to using REML to run an analysis.

Option: PRINT.
Parameters: none.

See also

Directives: REML, VCOMPONENTS, VSTRUCTURE, VRESIDUAL.
Genstat Reference Manual 1 Summary section on: REML analysis of linear mixed models.

VSTRUCTURE

Defines a variance structure for random effects in a REML model.

Options

TERMS = *formula*	Model terms for which the covariance structure is to be defined
FORMATION = *string token*	Whether the structure is formed by direct product construction or by definition of the whole matrix (direct, whole); default dire
CORRELATE = *string token*	Whether to impose correlation across the model terms if several are specified (none, positivedefinite, unrestricted); default none
CINITIAL = *scalars*	Initial values for covariance matrix across terms
COORDINATES = *matrix* or *variates*	Coordinates of the data points to be used in calculating distance-based models

Parameters

MODELTYPE = *string tokens*	Type of covariance model associated with the term(s), or with individual factors in the term(s) if FORMATION=direct (identity, fixed, AR, MA, ARMA, power, boundedlinear, circular, spherical, linearvariance, banded, correlation, antedependence, unstructured, diagonal, uniform, FA, FAequal) default iden
ORDER = *scalar*	Order of model
HETEROGENEITY = *string token*	Heterogeneity for correlation matrices (none, outside); default none
METRIC = *string token*	How to calculate distances when MODELTYPE=power (cityblock, squared, euclidean); default city
FACTOR = *factors*	Factors over which to form direct products
MATRIX = *symmetric matrices*, *diagonal matrices* or *pointers*	Defines matrix values for a term or the factors when MODELTYPE=fixed
INVERSE = *symmetric matrices*, *diagonal matrices* or *pointers*	Define values for matrix inverses (instead of the fixed matrices themselves) when MODELTYPE=fixed
DISTANCES = *symmetric matrices*	Symmetric matrix of pre-formed distances to be used in distance-based models of order one
COORDINATES = *matrices*, *variates* or *pointers*	Specifies coordinates of each factor level to be used in calculating distance-based models
INITIAL = *scalars*, *variates*, *matrices*, *symmetric matrices* or *pointers*	Initial parameter values for each correlation matrix (supplied in the structures appropriate for the model concerned)
CONSTRAINTS = *texts*	Texts containing strings none, fix or positive to define constraints for the parameters in each model
EQUALITYCONSTRAINTS = *variates*	Non-zero values in the variate indicate groups of parameters whose values are to be constrained to be equal

Description

VSTRUCTURE can be used to define the form of covariance structure for any term in the random model defined for REML by VCOMPONENTS. By default, the effects for each random term are assumed to be independent with common variance σ_j^2 for term j, that is, the random term has covariance matrix $\sigma_j^2 I$. VSTRUCTURE is used to define correlation between random effects within terms, to allow a changing variance within a term, and to define correlations between different random terms. These models are particularly useful when fitting linear models to repeated measurements or spatial data and for random coefficient regression.

VSTRUCTURE can only be used after VCOMPONENTS has been used to define the fixed and random models. It can be used more than once to define different structures for different random terms. The information is accumulated within Genstat, and it will all be used by subsequent REML commands. You can check on the model and covariance structures defined at any time by using the VSTATUS directive. To cancel a covariance structure for a term you simply need to use VSTRUCTURE to change the model back to the scaled identity matrix $\sigma_j^2 I$. To cancel all covariance structures you can give a new VCOMPONENTS command.

For a random term constructed from more than one factor, the covariance matrix can be formed either as a single matrix for the whole term, or as the direct product of several matrices corresponding to the factors. Consider an analysis of repeated measurements where data has been taken weekly from each subject, and one of several different treatments has been applied to each subject. It is likely that data taken from the same subject will be correlated, with correlation decreasing over time, but that subjects will be independent. This corresponds to an $I \otimes C$ covariance structure, where the identity matrix I corresponds to the independent subjects, and the covariance matrix C corresponds to the correlated measurements over time within subjects. If we take C to be an auto-regressive process of order 1, this can be defined and fitted as follows:

```
VCOMPONENTS [FIXED=Tmt] RANDOM=Subject.Week
VSTRUCTURE [TERM=Subject.Week] MODELTYPE=I,AR; ORDER=1; \
  FACTOR=Subject,Week
REML Y
```

The TERM option is used to specify the term to which the covariance structure is to be applied. For each factor in the term you can then specify the covariance model to be applied (see below for list of available models). However, it is not necessary to specify factors for which the default identity model is required, so the following is an equivalent specification:

```
VCOMPONENTS [FIXED=Tmt] RANDOM=Subject.Week
VSTRUCTURE [TERM=Subject.Week] MODELTYPE=AR; ORDER=1; FACTOR=Week
```

To cancel the covariance structure for the term, a null setting is sufficient:

```
VSTRUCTURE [TERM=Subject.Week]
```

Although the covariance structure for each term here is of the form $G_j = I$, the variance matrix for the data is of the form

$$V = \sigma^2 (\Sigma_j \gamma_j Z_j G_j Z_j' + I)$$

In this case the random subject term generates correlations that are equal across all the times within subjects. It is important to remember that including a random term in the model will generate uniform correlations between units with the same values of the random factor(s). Retaining these terms in the model as well as specifying a correlated structure may be appropriate for some data sets, but can sometimes lead to difficulties in parameter estimation.

The possible settings for the MODELTYPE parameter, generating symmetric covariance matrices C ($C_{i,j} = C_{j,i}$ for all i, j), are listed below. Where more than one model order can be used, the default is shown in bold and can be changed by using the ORDER option. For the AR, MA, ARMA, power and banded models, the order is the same as the number of parameters to be fitted. For the banded, correlation, ante-dependence and unstructured models, the order is the number of non-zero off-diagonal bands in the matrix. For the FAequal and FA

models, the order is the number of columns in the matrix Λ.

`identity`	identity matrix	$C_{i,i} = 1$, $C_{i,j} = 0$, for $i \neq j$
`fixed`	fixed matrix	$C_{i,j}$ specified
`AR`	auto-regressive order **1** or 2 (φ_2=0 for order 1)	$C_{i,i} = 1$ $C_{i+1,i} = \varphi_1 / (1-\varphi_2)$ $C_{i,j} = \varphi_1 C_{i-1,j} + \varphi_2 C_{i-2,j}$, $i > j+1$, $-1 < \varphi_1, \varphi_2 < 1$, $\lvert\varphi_1+\varphi_2\rvert<1$, $\varphi_2-\varphi_1<1$, $\varphi_2>-1$
`MA`	moving average order **1** or 2 (θ_2=0 for order 1)	$C_{i,i} = 1$ $C_{i+1,i} = -\theta_1(1-\theta_2)/(1+\theta_1^2+\theta_2^2)$ $C_{i+2,i} = -\theta_2 / (1+\theta_1^2+\theta_2^2)$ $C_{i,j} = 0$, $i>j+2$ $-1 < \theta_1, \theta_2 < 1$, $\theta_2 \pm \theta_1 < 1$
`ARMA`	auto-regressive moving-average order 1	$C_{i,i} = 1$ $C_{i+1,i} = (\theta-\varphi)(1-\varphi\theta)/(1+\theta^2-2\varphi\theta)$ $C_{i,j} = \varphi C_{i-1,j}$, $i>j+1$ $-1 < \varphi, \theta < 1$
`power`	based on distance order **1** or 2 ($\varphi_1 = \varphi_2$ for order 1)	$C_{i,i} = 1$ $C_{i,j} = \varphi_1^{d_1}\varphi_2^{d_2}$ d_1, d_2 = distance in 1st and 2nd dimensions $0 < \varphi_1, \varphi_2 < 1$
`boundedlinear`	based on distance order 1	$C_{i,j} = 1 - d/\varphi$ for $d \le \varphi$, $C_{i,j} = 0$ for $d > \varphi$ $0 < \varphi$
`circular`	based on distance order 1	$C_{i,j} = 1 - (2/\pi)\ \{(d/\varphi)\sqrt{(1-(d/\varphi)^2)} + \sin^{-1}(d/\varphi)\}$ for $d \le \varphi$, $C_{i,j} = 0$ for $d > \varphi$ $0 < \varphi$
`spherical`	based on distance order 1	$C_{i,j} = 1 - 1.5\ (d/\varphi) + 0.5\ (d/\varphi)^3$ for $d \le \varphi$, $C_{i,j} = 0$ for $d > \varphi$ $0 < \varphi$
`linearvariance`	based on distance order 1	$C_{i,j} = 1 - 2\varphi\ d / \max(d)$ $0 < \varphi < 1$
`banded`	equal bands **1** < order < nrows−1	$C_{i,i} = 1$ $C_{i+k,i} = \theta_k$, $1 < k <$ order $-1 < \theta_k < 1$ $C_{i+k,i} = 0$, otherwise
`correlation`	general correlation matrix 1 < order < **nrows−1**	$C_{i,i} = 1$ $C_{i,j} = \theta_{ij}$, $1 < \lvert i-j\rvert \le$ order $C_{i,j} = 0$, $\lvert i-j\rvert >$ order $-1 < \theta_{ij} < 1$
`uniform`	uniform matrix	$C_{i,j} = \theta$ for all i,j
`diagonal`	diagonal matrix	$C_{i,i} = \theta_i$ $C_{i,j} = 0$, $i \neq j$

`antedependence`	ante-dependence model **1** < order < nrows − 1	$C^{-1} = UD^{-1}U'$ $D_{i,i}{}^{-1} = d_i^{-1}$, $D_{i,j} = 0$ for $i \neq j$ $U_{i,i} = 1$, $U_{i,j} = u_{ij}$, $1 \leq j - i \leq$ order $U_{i,j} = 0$, for $i > j$
`unstructured`	general covariance matrix 1<order< **nrows**−**1**	$C_{i,j} = \theta_{ij}$, $0 < \lvert i-j \rvert \leq$ order $C_{i,j} = 0$, $\lvert i-j \rvert >$ order
`FA`	factor analytic order = **1** or 2	$C = \Lambda\Lambda' + \Psi$ Λ is an nrows × q matrix order=q $\Psi_i = \psi_i$ for i=1...nrows
`FAequal`	factor analytic with common variance order = **1** or 2	$C = \Lambda\Lambda' + \Psi$ Λ is an nrows × q matrix order=q $\Psi_i = \psi$ for i=1...nrows

Initial parameter values can be specified using the `INITIAL` parameter. For most models, the number of initial values required is the number of parameters, and default values will be generated. However, for `unstructured` models, a full covariance matrix of initial values must be given, and for the `correlation` model a full correlation matrix must be provided. For the `ante-dependence` model, either a full covariance matrix can be provided, or a pointer to a U and a D^{-1} matrix of the correct forms. For the `FA` and `FAequal` models, a pointer must be used to give the initial Λ and Ψ matrices, otherwise default initial values are generated. The `FAequal` model can be used to get initial values for the `FA` model. Initial values are required for these models because the algorithm may not converge when many parameters are fitted if the starting values are not realistic. Initial values might be generated from covariance matrices estimated by fitting simpler models, or from residuals from a null variance model. A missing value in the initial values is taken to mean that the value is inestimable and it will be fixed at a small value for the analysis. Alternatively, a parameter can be fixed at its initial value using the `CONSTRAINTS` parameter. The codes (not case sensitive and able to be abbreviated) may take value `fix` to indicate the parameter is to be fixed at its initial value, `positive` to indicate it is to remain positive or `none` to indicate no constraints. The default is a positive constraint or no constraint depending on context; for example scaling parameters are always constrained to remain positive. The `EQUALITYCONSTRAINTS` parameter allows you to constrain some of the parameters to have the same value. The variate that it specifies contains a zero value if there is no constraint, and an identical integer value for any set of parameters whose values are to be equal. So, a variate containing the values (0,1,2,1,2) would constrain the second parameter to be equal to the fourth parameter, and the third parameter to be equal to the fifth parameter.

It may sometimes be desirable to allow for unequal variances for the models defined in terms of correlation matrices: that is, for the `AR`, `MA`, `ARMA`, `uniform`, `power`, `boundedlinear`, `circular`, `spherical`, `linearvariance`, `banded` and `correlation` models. This can be done by setting option `HETEROGENEITY=outside`. This means a diagonal matrix D of standard errors will be applied to the correlation matrix C to generate a matrix $D^{½}CD^{½}$. In this case, a number of extra parameters (equal to the number of effects in the factor or term) should be added to the vector of initial values. These models allow investigation of a structured correlation pattern for changing variances and are particularly useful in the analysis of repeated measurements data when variance increases over time. For example, to allow for changing variance over time in our example above, we can specify

```
VCOMPONENTS [FIXED=Tmt] RANDOM=Subject.Week
VSTRUCTURE [TERM=Subject.Week] MODELTYPE=AR; ORDER=1;\
  FACTOR=Week; HETEROGENEITY=outside
REML Y
```

In some circumstances, you may wish to define a single model to apply to the whole term, instead of using the direct product form illustrated above. In this case, you should set option `FORM=whole`. Note that, when a term consists of a single factor, it is not necessary to set the `FACTOR` option.

With `MODELTYPE=fixed`, you must either use the `MATRIX` option to specify the values of the covariance matrix C, or the `INVERSE` option to specify the inverse matrix. The values of the matrix or its inverse can be supplied as diagonal matrices or symmetric matrices. In addition, values for the inverse matrix can be supplied in sparse form as a pointer. The output from `VPEDIGREE` is designed for input here, but you can also define the inverse matrix explicitly. The second element of the pointer should then be a variate containing the non-zero values of the inverse in lower triangular order. The first element should be a factor, with number of levels equal to the number of rows $n(n+1)/2$ of the matrix. This has firstly a block of n values giving the position in the variate of the first value stored for each row. There is then a block of values for each row in turn, giving the columns in which each non-zero value appears.

When `MODELTYPE=power` is used to define a distance-based model, the model can be of order 1 (isotropic) or 2 (anisotropic). For models with `ORDER=1`, a single set of distances must be formed. The necessary information can be supplied using either the `COORDINATES` option, or the `COORDINATES` parameter, or the `DISTANCES` parameter. With the `COORDINATES` option you can specify either a matrix, or a list of variates, to define multi-dimensional coordinates for each unit of the data. The length of the variates, or the number of rows of the matrix, must be equal to the number of data values. The number of variates, or the number of columns of the matrix, is equal to the number of dimensions. The coordinates for the levels of each `FACTOR` are then calculated as the mean values of the coordinates of the units included in the analysis with those levels. Alternatively, you can use the `COORDINATES` parameter to specify a single variate, a pointer to several variates or a matrix to define multi-dimensional coordinates for each level of the `FACTOR`. This parameter takes precedence over the `COORDINATES` option. The length of the variates, or the number of rows of the matrix, must be equal to the number of levels of the `FACTOR`. The number of variates, or the number of columns of the matrix, is again equal to the number of dimensions.

The distance calculation is defined by the `METRIC` option. For levels i and j with n-dimensional coordinates $\{c_{ik}: k=1...n\}$ and $\{c_{jk}: k=1...n\}$ the distance d_{ij} is defined as

$d_{ij} = \Sigma_k |c_{ik} - c_{jk}|$ for `METRIC=cityblock` (the default);
$d_{ij} = \Sigma_k (c_{ik} - c_{jk})^2$ for `METRIC=squared`; and
$d_{ij} = \{\Sigma_k (c_{ik} - c_{jk})^2\}^{1/2}$ for `METRIC=euclidean`.

Finally, you can supply a symmetric matrix of pre-calculated distances, using the `DISTANCES` parameter, and this takes precedence over the `COORDINATES` parameter and option. The number of rows of the `DISTANCES` matrix must be equal to the number of levels of the `FACTOR`.

When `MODELTYPE=power` and `ORDER=2`, the `DISTANCES` parameter cannot be used, and only two-dimensional coordinates are allowed. The coordinates must be specified using either the `COORDINATES` option or parameter, as described above. The distances are calculated within each dimension separately, according to the setting of the `METRIC` option. In this case the Euclidean and city-block distances are equivalent.

The spherical family of geostatistical models correspond to the `MODELTYPE` settings `boundedlinear` (for one-dimensional distances), `circular` (for one or two dimensions) and `spherical` (for one or two dimensions). For further details, see Webster & Oliver (2007). These models are based on distances, and require coordinates to be supplied using either the `COORDINATES` option (to give coordinates for each data value), or the `COORDINATES` parameter

(to give coordinates for each factor level), as described for `MODELTYPE=power` above. The parameter φ is interpreted as the range at which the correlation is considered to have decayed to zero. A small value therefore indicates weak correlation, and a large value indicates stronger correlation. These models do not have continuous second derivatives, and their log-likelihood may be multi-modal. To detect this potential problem, it is therefore important to start their estimation from several different initial values; this can be done using the `INITIAL` parameter as described above. To ensure that the estimated correlation matrix differs from the identity matrix, it is necessary for the range parameter to be larger than the minimum distance specified by the coordinates; any initial value smaller than this will be adjusted.

The setting `MODELTYPE=linearvariance` specifies the linear variance model of Williams (1986), extended by Piepho & Williams (2010). This model is parameterized so that the parameter φ lies in the range [0,1], which allows correlations in the range [-1,1]. Values of φ close to one indicate weak correlation and values close to zero indicate strong correlation between neighbouring observations.

The `CORRELATE` option allows you to specify correlations between model terms which have equal numbers of effects. A common correlation will then be fitted between parallel effects. For example, consider a random coefficient regression model where the fixed model contains common response to covariate `X` and the random model allows for deviations in the intercept and slope about this line for each subject. The random intercept and slope for each subject may be correlated, but subjects are independent. This correlation across terms is defined using the `CORRELATE` option as follows:

```
VCOMPONENTS [FIXED=X] RANDOM=SUBJECT+SUBJECT.X
VSTRUCTURE [SUBJECT+SUBJECT.X; CORRELATE=positivedefinite;\
  CINITIAL=!(1,0.1,0.3); FORM=whole]
```

The `CORRELATE` option setting `positivedefinite` is used to ensure that the correlation matrix between the terms remains positive definite. This constraint can be relaxed using the setting `unrestricted` (an unstructured covariance matrix is then used to describe covariance across the terms). The model fitting is done here in terms of a covariance matrix, where the diagonal elements are the gammas for the correlated terms. The `CINITIAL` option is used to give initial values for this matrix. If no initial values are given, the initial values are taken from initial gamma values given in `VCOMPONENTS` when the model is declared. When correlations are declared between terms, you must set `FORMATION=whole`. In the random coefficient regression model above, no correlation structure is declared within terms since the subjects are independent. However, it is possible to declare correlation/covariance models within terms as usual. For example, an animal breeding model might use `VPEDIGREE` to set up covariances within terms as follows:

```
VPEDIGREE animal; FEMALE=dam; MALE=sire; INVERSE=Ainv
VCOMPONENTS [FIXED=Trt] RANDOM=animal+dam+env
VSTRUCTURE [animal+dam; CORRELATE=unr; FORM=whole] \
  MODELTYPE=fixed; INVERSE=Ainv
```

These declarations set up random terms with covariance structures of the form:

$\mathrm{cov}(\texttt{animal}) = \sigma_a^2 A$, $\mathrm{cov}(\texttt{dam}) = \sigma_d^2 A$, $\mathrm{cov}(\texttt{animal}, \texttt{dam}) = \sigma_{ad} A$.

Direct Products

Although the direct product construction used to build the covariance structures does not generally constrain the models that can be fitted to any data set, you should be aware of the implications that arise when defining covariance structures for the residual term. The REML algorithm used by Genstat detects the presence of the residual term in the model by searching for terms with number of levels equal to the number of data values, n. When no covariance structures are specified, the first term with number of levels > n is used as the residual. However, when covariance structures are defined, the form of the variance model is

$$V = \sigma^2 (\Sigma_j \gamma_j Z_j G_j Z_j' + R)$$

where matrix *R* corresponds to the residual term and has *n* rows. For this reason, any term found with > *n* rows will not be used as the residual if it has a covariance matrix. If no valid residual term is found, a residual term will automatically be added to the model. This may result in an extra error term being fitted unintentionally. An example where this may happen is in repeated measurements data where unequal numbers of measurements have been taken on subjects. If direct product construction is used, the matrix generated will have more rows than the data and cannot be used as *R*. A work-around is to put missing values in the data set to give equal replication and use REML option MVINCLUDE=yvariate to retain the missing values in the analysis. Alternatively, you could fix the residual component at a small value.

Note that in the repeated measurements example above, if measurements are taken at different times for each subject, the direct product structure is not appropriate. In this case, a power model may be fitted over the whole term, constraining the between subject correlation to zero:

```
VSTRUCTURE [TERM=Subject.Week; FORM=whole;\
  COORD=subject,week] MODELTYPE=power; ORDER=2;\
  INITIAL=!(0,0.1); CONSTRAIN=!T(Fix,None)
```

Note that the parameters run in the order of the coordinates vectors (which are variate forms of the model factors).

Options: TERMS, FORMATION, CORRELATE, CINITIAL, COORDINATES.
Parameters: MODELTYPE, ORDER, HETEROGENEITY, METRIC, FACTOR, MATRIX, INVERSE, DISTANCES, COORDINATES, INITIAL, CONSTRAINTS, EQUALITYCONSTRAINTS.

References

Piepho, H.P. & Williams, E.R. (2010). Linear variance models for plant breeding trials. *Plant Breeding*, **129**, 1-8.

Webster, R. & Oliver, M.A. (2007). *Geostatistics for Environmental Scientists, 2nd edition*. Wiley, Chichester.

Williams, E.R. (1986). A neighbour model for field experiments. *Biometrika*, **73**, 279-87.

See also

Directives: REML, VCOMPONENTS, VRESIDUAL, VSTATUS.
Procedure: VFSTRUCTURE, VNEARESTNEIGHBOUR.
Genstat Reference Manual 1 Summary sections on: REML analysis of linear mixed models, Repeated measurements.

WORKSPACE

Accesses private data structures for use in procedures.

No options

Parameters

NAME = *texts*	Texts, each containing a single line, to give the names used to identify the private data structures
DUMMY = *identifiers*	Dummy structure to be used to refer to each private data structure

Description

The WORKSPACE directive is intended particularly for writers of procedures. It allows data to be accessed within a number of procedures, and in the main program if needed. You merely need to decide how to label your workspace "areas". Genstat reserves a data structure for each one, and WORKSPACE allows you to link this to a dummy (of your choice) within any procedure or in the outer program itself. For example

```
WORKSPACE 'AUNBALANCED work'; Wspace
TEXT    [VALUES=Yvar,Factopt] Wlabels
POINTER [NVALUES=Wlabels] Wspace
VARIATE Wspace['Yvar']
SCALAR  Wspace['Factopt']
```

names the area 'AUNBALANCED work' and sets the dummy Wspace to the associated data structure. The data structure is then defined to be a pointer with two values, the variate Wspace['Yvar'] and the scalar Wspace['Factopt']. A similar WORKSPACE statement can then be used later on (in another procedure) to access the same information. For example

```
WORKSPACE 'AUNBALANCED work'; Abwork
```

links the dummy Abwork to the pointer, allowing us to refer to Abwork['Yvar'] and Abwork['Factopt']. This will be used particularly within the Genstat Procedure Library, to link suites of associated procedures so, for safety, you should avoid prefixing the name of any workspace of your own by G5PL.

Options: none.
Parameters: NAME, DUMMY.

See also

Directives: PROCEDURE, DUMMY, POINTER.
Genstat Reference Manual 1 Summary section on: Program control.

XAXIS

Defines the x-axis in each window for high-resolution graphics.

Option

`RESET` = *string token*	Whether to reset the axis definition to the default values (`yes`, `no`); default `no`

Parameters

`WINDOW` = *scalars*	Numbers of the windows
`TITLE` = *texts*	Title for the axis
`TPOSITION` = *string tokens*	Position of title (`middle`, `end`)
`TDIRECTION` = *string tokens*	Direction of title (`parallel`, `perpendicular`)
`LOWER` = *scalars*	Lower bound for axis
`UPPER` = *scalars*	Upper bound for axis
`MARKS` = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along the axis (variate)
`MPOSITION` = *string tokens*	Positioning of the tick marks on the axis (`inside`, `outside`, `across`)
`LABELS` = *texts* or *variates*	Labels at each major tick mark
`LPOSITION` = *string tokens*	Position of the axis labels (`inside`, `outside`)
`LDIRECTION` = *string tokens*	Direction of the axis labels (`parallel`, `perpendicular`)
`LROTATION` = *scalars* or *variates*	Rotation of the axis labels
`NSUBTICKS` = *scalars*	Number of subticks per interval (ignored if `MARKS` is a variate)
`YORIGIN` = *scalars*	Position on y-axis at which the axis is drawn
`ZORIGIN` = *scalars*	Position on z-axis at which the axis is drawn
`PENTITLE` = *scalars*	Pen to use to write the axis title
`PENAXIS` = *scalars*	Pen to use to draw the axis
`PENLABELS` = *scalars*	Pen to use to write the axis labels
`ARROWHEAD` = *string tokens*	Whether the axis should have an arrowhead (`include`, `omit`)
`ACTION` = *string tokens*	Whether to display or hide the axis (`display`, `hide`)
`TRANSFORM` = *string tokens*	Transformed scale for the axis (`identity`, `log`, `log10`, `logit`, `probit`, `cloglog`, `square`, `exp`, `exp10`, `ilogit`, `iprobit`, `icloglog`, `root`); default `iden`
`LINKED` = *scalars*	Linked axis whose definitions should be used for this axis in 2-dimensional graphs; default `*` i.e. none
`MLOWER%` = *scalars*	How large a margin to set between the lowest x-value and the lower value of the axis, if not set explicitly by `LOWER` (expressed as a percentage of the range of the x-values)
`MUPPER%` = *scalars*	How large a margin to set between the largest x-value and the upper value of the axis, if not set explicitly by `UPPER` (expressed as a percentage of the range of the x-values)
`DECIMALS` = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
`DREPRESENTATION` = *scalars*, *variates* or *texts*	Format to use for dates and times printed at the marks
`VREPRESENTATION` = *string tokens*	Format to use for numbers printed at the marks

	(decimal, engineering, scientific); default deci
YOMETHOD = *string tokens*	Method to use to set the position of the origin on the y-axis if not set explicitly by YORIGIN (upper, lower, center, centre)
ZOMETHOD = *string tokens*	Method to use to set the position of the origin on the z-axis if not set explicitly by ZORIGIN (upper, lower, center, centre)
REVERSE = *string tokens*	Whether to reverse the axis direction to run from upper to lower instead of the default lower to upper (yes, no); default no
SAVE = *pointers*	Saves details of the current settings for the axis concerned

Description

There is a definition for the x-axis associated with each Genstat graphics window. This specifies how the x-axis is to be drawn when graphical output is produced in that window. The default definition for each axis requires some of the features to be determined from the data, as described below. Others have fixed defaults that are independent of the data. The XAXIS directive can be used to override the default action and specify particular aspects of the x-axis explicitly. (Similarly, directives YAXIS and ZAXIS modify the y- and z-axis definitions, respectively.) All the parameters of XAXIS are relevant when using DGRAPH, but for other directives only some of the parameters are used.

The WINDOW parameter specifies the window whose axis definition is to be altered. WINDOW can be set to a list of window numbers, in which case the other parameter lists are cycled in parallel, in the usual way. By default, only those aspects specified by subsequent parameter lists are modified; any parameters that are not set will retain their current settings. Alternatively, you can specify option RESET=yes to reset the values of any parameters that are not set for each window, back to the default values that are set up by Genstat at the start of a job.

The LOWER and UPPER parameters specify the lower and upper bounds for the axis. By default, Genstat derives suitable axis bounds from the data, as described for the appropriate directive. You can obtain an inverted scale by setting parameter REVERSE=yes. The values specified with these parameters are on the scale of the data values that are plotted, and are independent of the normalized device coordinates used to define the window size in FRAME. The MLOWER% parameter controls the size of margin that is provided between the lower value of the axis and the smallest x-value, if the lower axis value is not set explicitly by LOWER. This is expressed as a percentage of the range of the x-values, and has the initial default of 5%. Similarly the MUPPER% parameter controls the size of the upper margin.

The YORIGIN parameter determines the value on the y-axis through which the axis is drawn. If its value is outside the y-axis bounds, the upper or lower bound is adjusted so that the axis will extend up to the specified origin. This applies whether you have set the bounds explicitly or have left Genstat to calculate them from the data. If YORIGIN is not set, the YOMETHOD parameter can specify how the position should be determined: either at the upper value on the y-axis, or the lower value, or in the centre. The initial default (if neither of these parameters has been specified) is to put the axis at the bottom of the y-axis, which will be the lower value unless the scale is reversed. The ZORIGIN and ZOMETHOD parameters set the position of the origin on the z-axis in a similar way.

You can specify a title for the axis using the TITLE parameter. This is limited to a single line of characters. The TPOSITION parameter controls whether the title is placed in the middle or at the end of the axis, and the TDIRECTION parameter controls whether it is written parallel or perpendicular to the axis.

The axis is marked with a scale, determined automatically so that tick marks are evenly spaced

and positioned to give "round" numbers for the scale values. You can set the MARKS parameter to a scalar to define the increment between tick marks. For example, setting MARKS=1.5 with bounds 10 and 2 causes tick marks to appear at 2, 3.5, 5, 6.5, 8 and 9.5. The interval must be a positive number, irrespective of the values of the bounds. Alternatively, you can set MARKS to a variate (with more than one value) to specify the actual positions of the tick marks on the axis. Any values that lie outside the axis bounds are ignored. The scale values printed next to the tick marks use a format that is determined automatically from the values, but if you set MARKS to a variate it will use the number of decimals specified in the variate declaration. If MARKS is unset or set to a scalar, you can use the NSUBTICKS parameter to specify a number of "subticks" to be drawn between each of the (major) tick marks.

When you set MARKS, you can also use the LABELS parameter to specify a set of labels to print at the (major) axis marks, instead of the numbers. For example,

```
TEXT [VALUES=Mon,Tues,Wed,Thur,Fri,Sat,Sun] Day
VARIATE [VALUES=1...31] Month
XAXIS 1; MARKS=Month; LABELS=Day
```

The strings within the text are cycled if necessary, so the number of strings can be less than the number of tick marks. The DECIMALS parameter can set the number of decimal places to use if you are printing numbers at the marks. If the numbers represent dates or times, you can specify their formats using the DREPRESENTATION parameter (see the PRINT directive for details). By default, numbers are printed in decimal form. If you would prefer scientific format you can set parameter VREPRESENTATION=scientific; numbers are then printed as a decimal number with absolute value less than 10, followed by an exponent (e.g. 3.4567 E4 for 34567). Alternatively, you can set VREPRESENTATION=engineering to use engineering format; the decimal number then has an absolute value less than 10000, so the exponent is a multiple of 3 (e.g. 34.567 E3 for 34567). With scientific or engineering formats, the DECIMALS parameter sets the number of significant figures to use rather than the number of decimal places.

The MPOSITION parameter controls the positioning of the tick marks, which can be drawn on the inside or the outside of the axis, or can be drawn across the axis. With the outside setting, the tick marks are drawn towards the outside of the plot; that is below the axis if the axis is in the lower half of the plot, or above the axis if it is in the top half of the plot. The aim is then to position the tick marks away from the main part of the plot, so that they interfere with the plotted points as little as possible. With the inside setting, the marks are drawn on the opposite side (that is, to the inside of the plot), while the across setting draws them across the axis. Similarly, the positioning of the scale markings or labels is controlled by the LPOSITION parameter, with settings inside or outside. The LDIRECTION parameter controls whether the scale markings or labels are written parallel or perpendicular to the axis. Alternatively, you can use the LROTATION parameter to specify the direction of the labels more precisely, as a rotation in degrees from the horizontal (i.e. parallel) direction. If LROTATION is specified, any setting of LDIRECTION is ignored.

Setting MARKS=* will return to the default positioning of the tick marks. Setting LABELS=* will switch off any labels previously specified. Setting MPOSITION=* will switch off any tick marks, and setting LPOSITION=* or LDIRECTION=* will switch off any labels.

The TRANSFORM parameter allows you to transform the scale of the axis. The tick marks are still defined and labelled according to the original scale, but their physical positions on the graph are transformed. So, for example, with TRANSFORM=log10, the equal physical distance between 1 and 10 would be the same as the distance between 10 and 100. The settings are the same as the names of the equivalent Genstat functions, with the addition of exp10 for the antilog transformation (i.e. 10^x), and square for x^2.

There are three parameters to control the pens to be used to draw the axis. These are PENTITLE, PENAXIS and PENLABEL, specifying the pen for the title, the axis and the labelling, respectively. The initial default is to use pens −1, −2 and −3 in every window. These pens are

given negative numbers to allow them to be distinguished from the pens used for the contents of the plot. They are initially set up to use colour 1, line style 1, thickness 1, size 1 and font 1. You can thus control which pens are used for drawing the axis in each window, and the attributes of those pens. For example, if no XAXIS statement has yet been given,

```
PEN -1; LINESTYLE=4; COLOUR=2
```

will request that the titles in every window should be written in line style 4 and colour 2; while

```
PEN 29; LINESTYLE=3; COLOUR=4
XAXIS 1; PENAXIS=29
```

will change the appearance of just the x-axis in window 1, as pen 29 is not used for the other windows. You should of course be careful of side-effects when changing the pen numbers. For example, pen 29 may also have been modified for use in a DGRAPH statement and other attributes may have been set that are not wanted when drawing the axis. You must use the RESET option if you want to restore these pen numbers to the initial defaults. (Genstat does not allow you to set negative pen numbers explicitly.)

The ARROWHEAD parameter controls whether the axis is drawn with an arrowhead at the end.

You may sometimes wish to use the axis definitions merely to control the positioning of the plot in the x-direction (using the UPPER and LOWER parameters), or you may wish to hide the axis temporarily in case it is obscuring information in the plot. You can do this by setting parameter ACTION=hide.

Axis annotation is plotted in the margins specified by the FRAME directive. You may wish to reduce the size of these margins if you have defined axes that use less space, for example by keeping within the area of the graph itself, or by omitting titles or labels. Space can thus be regained and used for plotting data. However, if the margins are too small the axis annotation may be "clipped" at the boundaries of the margins; if this happens, you can use FRAME to increase the margin size. The margins are used by DGRAPH, DHISTOGRAM and DCONTOUR, but they are ignored by other directives.

The LINKED parameter is useful when you have several related plots in different windows within the frame. If, for example, you set LINKED=n, the attributes of the current x-axis will all be taken (at the time of plotting) from the definition of the x-axis for any 2-dimensional graph in window *n*. Also, you can edit the attributes of all the linked axes simultaneously in the graphics viewer in Genstat *for Windows*.

The current settings of the axis for a particular window can be saved in a pointer supplied by the SAVE parameter. The elements of the pointer are labelled to identify the components. The settings are those for the axis itself, so you should check that the axis is not linked to one in another window. (The 'linked' element contains the window number, or a missing value there is no link.) The SAVE parameter is of most use within procedures, where it may be necessary to check or modify particular XAXIS settings before constructing complicated graphs. Also, the DKEEP directive allows you to extract the actual bounds used when plotting; these will be the bounds determined from the data if none have been defined explicitly by XAXIS.

Option: RESET.

Parameters: WINDOW, TITLE, TPOSITION, TDIRECTION, LOWER, UPPER, MARKS, MPOSITION, LABELS, LPOSITION, LDIRECTION, LROTATION, NSUBTICKS, YORIGIN, ZORIGIN, PENTITLE, PENAXIS, PENLABELS, ARROWHEAD, ACTION, TRANSFORM, LINKED, MLOWER%, MUPPER%, DECIMALS, DREPRESENTATION, VREPRESENTATION, YOMETHOD, ZOMETHOD, SAVE.

See also
Directives: YAXIS, ZAXIS, AXIS, FRAME.
Procedure: DHELP.
Genstat Reference Manual 1 Summary section on: Graphics.

YAXIS

Defines the y-axis in each window for high-resolution graphics.

Option

RESET = *string token*	Whether to reset the axis definition to the default values (yes, no); default no

Parameters

WINDOW = *scalars*	Numbers of the windows
TITLE = *texts*	Title for the axis
TPOSITION = *string tokens*	Position of title (middle, end)
TDIRECTION = *string tokens*	Direction of title (parallel, perpendicular)
LOWER = *scalars*	Lower bound for axis
UPPER = *scalars*	Upper bound for axis
MARKS = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along the axis (variate)
MPOSITION = *string tokens*	Positioning of the tick marks on the axis (inside, outside, across)
LABELS = *texts* or *variates*	Labels at each major tick mark
LPOSITION = *string tokens*	Position of the axis labels (inside, outside)
LDIRECTION = *string tokens*	Direction of the axis labels (parallel, perpendicular)
LROTATION = *scalars* or *variates*	Rotation of the axis labels
NSUBTICKS = *scalars*	Number of subticks per interval (ignored if MARKS is a variate)
XORIGIN = *scalars*	Position on x-axis at which the axis is drawn
ZORIGIN = *scalars*	Position on z-axis at which the axis is drawn
PENTITLE = *scalars*	Pen to use to write the axis title
PENAXIS = *scalars*	Pen to use to draw the axis
PENLABELS = *scalars*	Pen to use to write the axis labels
ARROWHEAD = *string tokens*	Whether the axis should have an arrowhead (include, omit)
ACTION = *string tokens*	Whether to display or hide the axis (display, hide)
TRANSFORM = *string tokens*	Transformed scale for the axis (identity, log, log10, logit, probit, cloglog, square, exp, exp10, ilogit, iprobit, icloglog, root); default iden
LINKED = *scalars*	Linked axis whose definitions should be used for this axis in 2-dimensional graphs; default * i.e. none
MLOWER% = *scalars*	How large a margin to set between the lowest y-value and the lower value of the axis, if not set explicitly by LOWER (expressed as a percentage of the range of the y-values)
MUPPER% = *scalar*s	How large a margin to set between the largest y-value and the upper value of the axis, if not set explicitly by UPPER (expressed as a percentage of the range of the y-values)
DECIMALS = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
DREPRESENTATION = *scalars*, *variates* or *texts*	Format to use for dates and times printed at the marks
VREPRESENTATION = *string tokens*	Format to use for numbers printed at the marks

	(`decimal`, `engineering`, `scientific`); default `deci`
`XOMETHOD` = *string tokens*	Method to use to set the position of the origin on the x-axis if not set explicitly by `XORIGIN` (`upper`, `lower`, `center`, `centre`)
`ZOMETHOD` = *string tokens*	Method to use to set the position of the origin on the z-axis if not set explicitly by `ZORIGIN` (`upper`, `lower`, `center`, `centre`)
`REVERSE` = *string tokens*	Whether to reverse the axis direction to run from upper to lower instead of the default lower to upper (`yes`, `no`); default `no`
`SAVE` = *pointers*	Saves details of the current settings for the axis concerned

Description

There is a definition for the y-axis associated with each Genstat graphics window. This specifies how the y-axis is to be drawn when graphical output is produced in that window. The default definition for each axis requires some of the features to be determined from the data. Others have fixed defaults that are independent of the data. The `YAXIS` directive can be used to override the default action and specify particular aspects of the y-axis explicitly. (Similarly, directives `XAXIS` and `ZAXIS` modify the x- and z-axis definitions, respectively.) All the parameters of `YAXIS` are relevant when using `DGRAPH`, but for other directives only some of the parameters are used. The syntax of `YAXIS` is identical to that of `XAXIS`, except that `YAXIS` has an `XORIGIN` parameter which replaces the `YORIGIN` parameter of `XAXIS`.

The `WINDOW` parameter specifies the window whose axis definition is to be altered. `WINDOW` can be set to a list of window numbers, in which case the other parameter lists are cycled in parallel, in the usual way. By default, only those aspects specified by subsequent parameter lists are modified; any parameters that are not set will retain their current settings. Alternatively, you can specify option `RESET=yes` to reset the values of any parameters that are not set for each window, back to the default values that are set up by Genstat at the start of a job.

As in `XAXIS`, the `LOWER`, `UPPER`, `MLOWER%` and `MUPPER%` parameters can specify the lower and upper bounds for the axis, the `REVERSE` parameter can reverse the axis, and the `TITLE`, `TPOSITION` and `TDIRECTION` parameter can define a title for the axis.

The `XORIGIN` parameter determines the value on the x-axis through which the axis is drawn. If its value is outside the x-axis bounds, the upper or lower bound is adjusted so that the axis will extend up to the specified origin. This applies whether you have set the bounds explicitly or have left Genstat to calculate them from the data. If `XORIGIN` is not set, the `XOMETHOD` parameter can specify how the position should be determined: either at the upper value on the x-axis, or the lower value, or in the centre. The initial default (if neither of these parameters has been specified) is to put the axis at the left-hand end, which will be the lower value unless the scale is reversed. The `ZORIGIN` and `ZOMETHOD` parameters set the position of the origin on the z-axis in a similar way, with the initial default that the axis is at the bottom of the z-axis.

The `MARKS`, `NSUBTICKS`, `LABELS`, `DECIMALS`, `DREPRESENTATION` and `VREPRESENTATION` parameters also operate as in `XAXIS`, to specify the markings on the axis, and their associated labels. The `MPOSITION`, `LPOSITION`, `LDIRECTION` and `LROTATION` parameters again control the positioning of the tick marks and labels. For a y-axis, the `outside` setting implies that the tick marks are drawn to the left of the axis if the axis is on the left-half side of the plot, or to the right of the axis if it is on the right-hand side. As in `XAXIS`, the `TRANSFORM` parameter allows you to transform the physical scale of the axis on the graph.

The `ARROWHEAD` parameter again controls whether the axis is drawn with an arrowhead at the end, and parameters `PENTITLE`, `PENAXIS` and `PENLABEL` specify the pen to be used for the title, the axis and the labelling, respectively. `ACTION` allows you to hide the axis, `LINKED` allows you

to take all the axis settings from a (linked) axis in another window, and SAVE allows you to save the current settings defined for the axis. Further details are given in the description of XAXIS.

Option: RESET.
Parameters: WINDOW, TITLE, TPOSITION, TDIRECTION, LOWER, UPPER, MARKS, MPOSITION, LABELS, LPOSITION, LDIRECTION, LROTATION, NSUBTICKS, XORIGIN, ZORIGIN, PENTITLE, PENAXIS, PENLABELS, ARROWHEAD, ACTION, TRANSFORM, LINKED, MLOWER%, MUPPER%, DECIMALS, DREPRESENTATION, VREPRESENTATION, XOMETHOD, ZOMETHOD, SAVE.

See also
Directives: XAXIS, ZAXIS, AXIS, FRAME.
Procedure: DHELP.
Genstat Reference Manual 1 Summary section on: Graphics.

ZAXIS

Defines the z-axis in each window for high-resolution graphics.

Option

`RESET` = *string token*	Whether to reset the axis definition to the default values (`yes`, `no`); default `no`

Parameters

`WINDOW` = *scalars*	Numbers of the windows
`TITLE` = *texts*	Title for the axis
`TPOSITION` = *string tokens*	Position of title (`middle`, `end`)
`TDIRECTION` = *string tokens*	Direction of title (`parallel`, `perpendicular`)
`LOWER` = *scalars*	Lower bound for axis
`UPPER` = *scalars*	Upper bound for axis
`MARKS` = *scalars* or *variates*	Distance between each tick mark (scalar) or positions of the marks along the axis (variate)
`MPOSITION` = *string tokens*	Positioning of the tick marks on the axis (`inside`, `outside`, `across`)
`LABELS` = *texts* or *variates*	Labels at each major tick mark
`LPOSITION` = *string tokens*	Position of the axis labels (`inside`, `outside`)
`LDIRECTION` = *string tokens*	Direction of the axis labels (`parallel`, `perpendicular`)
`LROTATION` = *scalars* or *variates*	Rotation of the axis labels
`NSUBTICKS` = *scalars*	Number of subticks per interval (ignored if `MARKS` is a variate)
`XORIGIN` = *scalars*	Position on x-axis at which the axis is drawn
`YORIGIN` = *scalars*	Position on y-axis at which the axis is drawn
`PENTITLE` = *scalars*	Pen to use to write the axis title
`PENAXIS` = *scalars*	Pen to use to draw the axis
`PENLABELS` = *scalars*	Pen to use to write the axis labels
`ARROWHEAD` = *string tokens*	Whether the axis should have an arrowhead (`include`, `omit`)
`ACTION` = *string tokens*	Whether to display or hide the axis (`display`, `hide`)
`MLOWER%` = *scalars*	How large a margin to set between the lowest z-value and the lower value of the axis, if not set explicitly by `LOWER` (expressed as a percentage of the range of the z-values)
`MUPPER%` = *scalars*	How large a margin to set between the largest z-value and the upper value of the axis, if not set explicitly by `UPPER` (expressed as a percentage of the range of the z-values)
`DECIMALS` = *scalars* or *variates*	Number of decimal places to use for numbers printed at the marks
`DREPRESENTATION` = *scalars*, *variates* or *texts*	Format to use for dates and times printed at the marks
`VREPRESENTATION` = *string tokens*	Format to use for numbers printed at the marks (`decimal`, `engineering`, `scientific`); default `deci`
`XOMETHOD` = *string tokens*	Method to use to set the position of the origin on the x-axis if not set explicitly by `XORIGIN` (`upper`, `lower`, `center`, `centre`)
`YOMETHOD` = *string tokens*	Method to use to set the position of the origin on the y-

	axis if not set explicitly by YORIGIN (upper, lower, center, centre)
REVERSE = *string tokens*	Whether to reverse the axis direction to run from upper to lower instead of the default lower to upper (yes, no); default no
SAVE = *pointers*	Saves details of the current settings for the axis concerned

Description

There is a definition for the z-axis associated with each Genstat graphics window. This specifies how the z-axis is to be drawn when three-dimensional graphical output is produced in that window. The default definition for each axis requires some of the features to be determined from the data. Others have fixed defaults that are independent of the data. The ZAXIS directive can be used to override the default action and specify particular aspects of the z-axis explicitly. (Similarly, directives XAXIS and YAXIS modify the x- and y-axis definitions, respectively.) All parameters of ZAXIS are relevant when using D3GRAPH, but for other directives only some of the parameters are used. The syntax of ZAXIS is identical to that of XAXIS, except that ZAXIS has an XORIGIN parameter instead of the ZORIGIN parameter of XAXIS.

The WINDOW parameter specifies the window whose axis definition is to be altered. WINDOW can be set to a list of window numbers, in which case the other parameter lists are cycled in parallel, in the usual way. By default, only those aspects specified by subsequent parameter lists are modified; any parameters that are not set will retain their current settings. Alternatively, you can specify option RESET=yes to reset the values of any parameters that are not set for each window, back to the default values that are set up by Genstat at the start of a job.

As in XAXIS, the LOWER, UPPER, MLOWER% and MUPPER% parameters can specify the lower and upper bounds for the axis, the REVERSE parameter can reverse the axis, and the TITLE, TPOSITION and TDIRECTION parameter can define a title for the axis.

The XORIGIN parameter determines the value on the x-axis through which the axis is drawn. If its value is outside the x-axis bounds, the upper or lower bound is adjusted so that the axis will extend up to the specified origin. This applies whether you have set the bounds explicitly or have left Genstat to calculate them from the data. If XORIGIN is not set, the XOMETHOD parameter can specify how the position should be determined: either at the upper value on the x-axis, or the lower value, or in the centre. The initial default (if neither of these parameters has been specified) is to put the axis at the lower end, which will be the lower value unless the scale is reversed. The YORIGIN and YOMETHOD parameters set the position of the origin on the y-axis in a similar way, with the initial default that the axis is at the bottom of the y-axis.

As in XAXIS, the MARKS, NSUBTICKS, LABELS, DECIMALS, DREPRESENTATION and VREPRESENTATION parameters specify the markings on the axis, and their associated labels. The MPOSITION, LPOSITION, LDIRECTION and LROTATION parameters control the positioning of the tick marks and labels. The ARROWHEAD parameter again controls whether the axis is drawn with an arrowhead at the end, and parameters PENTITLE, PENAXIS and PENLABEL specify the pen to be used for the title, the axis and the labelling, respectively. ACTION allows you to hide the axis, and SAVE allows you to save the current settings defined for the axis. Full details of all these parameters are given in the description of XAXIS.

Option: RESET.

Parameters: WINDOW, TITLE, TPOSITION, TDIRECTION, LOWER, UPPER, MARKS, MPOSITION, LABELS, LPOSITION, LDIRECTION, LROTATION, NSUBTICKS, XORIGIN, YORIGIN, PENTITLE, PENAXIS, PENLABELS, ARROWHEAD, ACTION, MLOWER%, MUPPER%, DECIMALS, DREPRESENTATION, VREPRESENTATION, XOMETHOD, YOMETHOD, SAVE.

See also
Directives: XAXIS, YAXIS, AXIS, FRAME.
Procedure: DHELP.
Genstat Reference Manual 1 Summary section on: Graphics.

%CD

Changes current directory, PC Windows only.

No options

Parameters

DIRECTORY = *text*	Directory to change to
CURRENT = *text*	Saves new directory

Description

The %CD directive can be used to change directory under Windows or DOS. The DIRECTORY specifies the directory as a text containing either the absolute or relative pathname, for example 'C:/CONSULT/DATA' or '../../PROJECT'.

Note that a forward slash (/) may be used as a directory separator character, as the backwards slash (\) is the Genstat continuation character and would have to be doubled up. Environment variables may be used in the path name, for example '%TEMP%/temp.out', and %GENDIR% can be used to specify the Genstat root directory (eg. C:).

The CURRENT parameter saves the new directory name. You can obtain the name of the current directory by typing the command

```
%CD '.'; CURRENT=curdir
```

which will set up curdir as a text containing the current directory name.

See also

Directive: SUSPEND.
Procedure: DIRECTORY.

%OPEN

Open a binary file for use with %WRITE.

No options

Parameter

NAME = *text*	Name of file to be opened for binary output using %WRITE

Description

%OPEN has a single parameter, NAME, which specifies the name of a file to be opened in binary mode for output by the %WRITE command. If the file already exists, it is deleted, and then reopened as a new file. The file can be used to communicate data to other programs that read in data in binary form. For example, it is used by FSPREADSHEET to write spreadsheet files.

The file should be closed by the %CLOSE directive when output has been completed. The %FPOSITION command can be used to query the current position in the file.

Options: none.
Parameter: NAME.

See also

Directives: %CLOSE, %FPOSITION, %WRITE.
Genstat Reference Manual 1 Summary section on: Input and output.

%FPOSITION

Returns the current position in the binary file opened by %OPEN.

No options

Parameter

scalar	Number of bytes of the current position from the start of the file

Description

%FPOSITION has a single unnamed parameter, which saves the current write position in the binary file opened by %OPEN. This can be used to relocate the %WRITE directive to that position in the file. For example, if you wanted to change a value in the file later during its use, you could save the position of that value, and then use %WRITE to write the new value into the file.

For example:

```
%FPOSITION file_length_position
%WRITE 0 "Write a dummy value to hold the position"
"... more %WRITE commands"
%FPOSITION file_length "Get current position = file length"
%WRITE [POSITION=file_length_position] file_length
```

Options: none.
Parameter: unnamed.

See also

Directives: %CLOSE, %OPEN, %WRITE.
Genstat Reference Manual 1 Summary section on: Input and output.

%LOG

Adds text into the Input Log window in the Genstat client.

No options

Parameter

text	Text to display in the Input Log window

Description

%LOG has a single unnamed parameter, which provides a text to be inserted at the end of the Input Log window in the Genstat client. This can be used to include commands or comments. For example,

```
%LOG '""Calculate the logs of the yields""'
%LOG 'CALCULATE logYield = LOG(Yield)'
```

You might want to do this to provide information about the actions of menu or an interactive procedure. For example, the STATEMENT parameter of the DESIGN procedure can save a command that could be used to form the design that has been generated. You might want to use %LOG to record this in the Input Log.

%LOG has no effect when used in GenBatch, and is available only in Windows versions of Genstat.

Options: none.
Parameter: unnamed.

See also

Directive: %MESSAGEBOX.
Genstat Reference Manual 1 Summary section on: Input and output.

%MESSAGEBOX

Display text in a dialog in the Genstat client.

Options

TITLE = *text*	Title for the dialog; default 'Genstat'
ICON = *string token*	Icon to display in the dialog (information, warning, error, question); default info

Parameter

text	text to display in the dialog

Description

%MESSAGEBOX has a single unnamed parameter, which provides a text to be displayed in a client dialog. The dialog remains on the screen until the user clicks its OK button. For example,

```
%MESSAGEBOX 'Some analyses were unable to calculate F statistics.'
```

This provides a good way to communicate warnings or notes to the user. For example, if faults or warnings have been disabled, it can be used to display just those faults or warnings that you want the user to see. It has no effect when used in GenBatch, and is available only in Windows versions of Genstat.

The TITLE option provides the title for the dialog. The default is to display Genstat. The ICON option specifies the icon to display on the left-hand side. The default is to display the information icon (i).

Options: TITLE, ICON.
Parameter: unnamed.

See also

Directive: %LOG.
Genstat Reference Manual 1 Summary section on: Input and output.

%OPEN

Open a binary file for use with %WRITE.

No options

Parameter

NAME = *text*	Name of file to be opened for binary output using %WRITE

Description

%OPEN has a single parameter, NAME, which specifies the name of a file to be opened in binary mode for output by the %WRITE command. If the file already exists, it is deleted, and then reopened as a new file. The file can be used to communicate data to other programs that read in data in binary form. For example, it is used by FSPREADSHEET to write spreadsheet files.

The file should be closed by the %CLOSE directive when output has been completed. The %FPOSITION command can be used to query the current position in the file.

Options: none.
Parameter: NAME.

See also

Directives: %CLOSE, %FPOSITION, %WRITE.
Genstat Reference Manual 1 Summary section on: Input and output.

%SLEEP

Pauses execution of the server for a time specified in seconds.

No options

Parameter

scalar	specifies the time in seconds to pause

Description

`%SLEEP` has a single, unnamed, parameter, which is a scalar giving the number of seconds for the server to halt processing. The current set of commands continues after this period of time. For example,

```
%SLEEP 2
```

stops the server processing for two seconds. This can be used to allow other processes to complete before the server resumes execution. For example, the saving of graphs to PNG files using the `OPEN` and `DEVICE` commands runs asynchronously to the server process that generates them. You may therefore need to wait for these to be created if they are to be used in another part of the program.

Options: none.
Parameter: unnamed.

%TEMPFILE

Creates a unique temporary file in the Genstat temporary folder.

No options

Parameters

PREFIX = *string*	Prefix for the filename
FILENAME = *text*	Saves the filename
INDEX = *scalar*	Saves the index number that follows the prefix in the filename

Description

%TEMPFILE creates a unique file in the Genstat temporary folder. The location of the temporary folder can be obtained by the GETTEMPFOLDER procedure, and is set by the %TMP% or %TEMP% environment variables on your computer. An empty file is created, so that other programs will not use the same name.

The PREFIX parameter provides the text to be used at the start of the filename. The extension of the filename is always .tmp. The unique filename is returned by the FILENAME parameter, and the INDEX parameter saves its index number. Note, the filename has the index value written in hexadecimal format, but the index is saved as an ordinary number. For example:

```
%TEMPFILE PREFIX='Out'; FILENAME=TFile; INDEX=filenumber
```

could return TFile as 'C:/Temp/Genstat/Out1F.tmp' and INDEX as 31 (31 in hexadecimal = 1F).

The file is not deleted automatically at the end of the Genstat run. So, when you have finished using it, you need to delete it yourself (for example, with the FDELETE directive).

Options: none.
Parameter: PREFIX, FILENAME, INDEX.

See also

Directives: CLOSE, FDELETE, OPEN.
Procedure: GETTEMPFOLDER.
Genstat Reference Manual 1 Summary section on: Input and output.

%WRITE

Writes values of data structures to a binary file opened by %OPEN.

Options

SEPARATOR = *scalar or text*	Separator character as a literal character or a scalar giving an ASCII code (0-255); default * i.e. none
TERMINATOR = *string token*	Terminator to use at the end of a text (null, newline) default null
POSITION = *scalar*	File position at which to write the data; default 0 writes at the current position

Parameters

DATA = *texts, scalars, variates or matrices*	Data structures to write to the file
FORMATTED = *string tokens*	Output format to use when writing the structures (bit, byte, shortint, longint, real, double, string, text, rawtext, factor); default depends on the type of data structure
NBYTES = *scalars*	Saves the number of bytes written to the file

Description

%WRITE writes values of data structures to the binary file opened by %OPEN. The data structures are specified by the DATA parameter.

The FORMATTED parameter specifies the format to be used to write the structures to the file. The available formats are:

bit	the numbers (0,1) in the structure are combined as single bits into bytes (e.g. 1,0,0,1,0,1,0,1,0,1,0,0,1,1,1,1 would give the two bytes 0x41CF)
byte	the numbers are written as a single byte (0 to 255)
shortint	the numbers are written as a 2-byte signed integer (-31767 to 32767)
longint	the numbers are written as 4-byte signed integer; default for a scalar
real	the numbers are written as single precision (REAL*4) values (missing value = –1e37)
double	the numbers are written as double precision (REAL*8) values (missing value = –1e307); default for a variate or matrix
string	the texts are written with a single byte giving the length, followed by the ASCII bytes in UTF8 format
text	the texts are written with 4 bytes giving the length, followed by the ASCII bytes in UTF8 format, default for a text
rawtext	the texts are written as ASCII bytes in UTF8 format
factor	the values are compressed into as few bytes as possible; the default for a factor

The factor format can be used only by factors. The number of bytes per item depends on the number of levels in the factor. For 1-3 levels, 4 items are combined per byte, for 4-15 levels, 2 items are combined per byte, for 16-255 levels, each item is written in a single byte, for 256-65535 levels, each item is written in a two bytes, otherwise each item is written in 4 bytes. A value of 0 is used to represent a missing value in the factor. The formats string, text and

rawtext can be used only by texts.

The SEPARATOR and TERMINATOR options are used only for FORMAT=rawtext. The SEPARATOR option can specify a single ASCII character to be used between items. By default there is no separator. The TERMINATOR option specifies what to write after each string. The default setting null uses a null character (byte 0 as in C strings). With the newline setting, the two newline characters (bytes 13 and 10) are use to terminate strings.

The POSITION option can be used to reset the position where the values are written. The default of 0 writes the values at the current position in the file.

The NBYTES option saves the number of bytes that have been written to the file.

Options: SEPARATOR, TERMINATOR, POSITION.
Parameters: DATA, FORMATTED, NBYTES.

Action with RESTRICT
%WRITE ignores restrictions on DATA.

See also
Directives: %OPEN, %CLOSE, %FPOSITION.
Genstat Reference Manual 1 Summary section on: Input and output.

Index