Biometris GenStat Procedure Library Manual
15th Edition

Edited by
Paul W. Goedhart & Jac T.N.M. Thissen

Biometris report 21.06.12 June 2012

Biometris
quantitative methods brought to life

Biometris GenStat Procedure Library Manual 15th Edition

Edited by

Paul W. Goedhart & Jac T.N.M. Thissen

Biometris is the integration of the Centre for Biometry of Plant Research International and the Department of Mathematical and Statistical Methods of Wageningen University. Biometris, part of Wageningen University and Research center (Wageningen UR), was established on 20th June 2001.

For more information please visit the website www.wageningenur.nl/biometris or contact:

Biometris, Wageningen UR
P.O. Box 100
6700 AC Wageningen, The Netherlands
Visiting address:
Buildingnumber 116
Bornsesteeg 47
6708 PD Wageningen
Phone: +31 (0)317 484085; Fax: +31 (0)317 483554
E-mail: Biometris@wur.nl

June 2012

Contents of the Biometris Procedure Library 15th Edition

Continue on next page

Contents continued

Disclaimer for the Biometris Procedure Library 15th Edition

Notes for the 15th Edition

- The Biometris GenStat Procedure Library and the library manual can be downloaded from the internet adress http://www.wageningenur.nl/en/show/GenStat-Procedures.htm. Just run the installation program to install the library.
- The Library contains a collection of procedures mainly written by members of Biometris of Wageningen UR. The library supplements the official GenStat Procedure Library which is distributed with GenStat itself. The Biometris Library is distributed over the Dutch agricultural research sites and is installed as a User Procedure Library. The Biometris procedures can therefore be used in exactly the same way as the standard GenStat directives and procedures.
- The `BIOMETRIS [PRINT=allfiles]` command outputs the example and source code of all Biometris procedures in a directory named "BiometrisSource" below the current working directory.
- Uncertainty and regression-based sensitivity analysis of a deterministic model can be performed using Biometris procedures `EDCONTINUOUS`, `GMULTIVARIATE`, `GUNITCUBE` and `RUNCERTAINTY`. Biometris report 11.12.05 describes these procedures in detail and includes a number of illustrative examples. This report is distributed with this library and can be found in the Biometris subdirectory below the AddIns folder of GenStat.
- Procedures `COMPACT`, `FDERIVATIVE`, `FINTEGRATE`, `FMINIMIZE`, `FROOTFIND`, `GAUSSPOINTS`, `GROBINOMIAL`, `GROPOISSON`, `QREGISTRY`, `R2NEGBINOMIAL`, `RLOGITNORMAL`, `SETDEVICE` and `SYM2VARIATE` are new in this release.
- Procedures `PPAIR` and `SETDEVICE` succeed a procedure in the official GenStat Procedure Library and have some new options and/or parameters.
- Procedures `BIOMETRIS`, `GENBATCH`, `PER2MUTE`, `QTIMEDELAY`, `RBETABINOMIAL` and `SFILENAME` have some new options and/or parameters.
- Procedures `PER2MUTE`, `RLMS`, `RPLS` and `RSELECT` use an external C# executable, using .NET Framework 4, which is called by means of the `SUSPEND` directive. Previously `PASS` was used to call an external Fortran DLL. The Fortran code was translated to C#. This might imply some very slight changes in output due to different numerical precision. Procedure `QREGISTRY` employs the same C# executable. Note that the switch from Fortran to C# induced some changes in the options and parameters of the `BIOMETRIS` procedure.
- Procedures `DIRLIST`, `GENBATCH`, `QDIRECTORY`, `QFILENAME`, `QMESSAGE`, `QPICKLIST`, `QSTOPWATCH`, `QTEXT`, `QTIMEDELAY` and `QYESNO` use an external WinBatch program to interact with the user. This program (BiometrisWinbatch.exe) is called by means of `SUSPEND [CONTINUE=no]`, The internet site www.winbatch.com provides more information about WinBatch.

BICORRELATE

J.T.N.M. Thissen

Forms pairwise correlations between variates including as many units as possible

contents - next

Options

PRINT = *string tokens*	What to print (correlations, nobservations, tests); default correlations
METHOD = *string token*	Type of test to make (against zero) for the correlations (twosided, greaterthan, lessthan); default twosided
CORRELATIONS = *symmetric matrix*	Stores the pairwise correlations between the variates specified by the VARIATES parameter
PROBABILITIES = *symmetric matrix*	Saves the test probabilities
NOBSERVATIONS = *symmetric matrix*	Stores the pairwise number of observations on which the correlations are based

Parameters

VARIATES = *variates*	Variates for which the correlations are to be calculated; must be set

Description

Procedure BICORRELATE calculates pairwise correlations by excluding only units with missing values in the corresponding pair of variates. Note that the CORRELATE directive and procedure FCORRELATION exclude all units with at least one missing value in the set of variates. Printed output is controlled by the PRINT option with settings:

correlations	prints the correlation matrix;
tests	prints tests for the correlations.

By default PRINT=correlation. The METHOD option indicates the type of test to be done, with settings:

twosided	for a two-sided test of the null hypothesis that the correlation is zero;
greaterthan	for a one-sided test of the null hypothesis that the correlation is not greater than zero;
lessthan	for a one-sided test of the null hypothesis that the correlation is not less than zero.

Tests cannot be produced if there are fewer than two observations. The correlation matrix can be saved using the CORRELATIONS option, the (symmetric) matrix of test probabilities can be saved using the PROBABILITIES option, and the number of observations upon which it is based can be saved using the NOBSERVATIONS option.

Method

BICORRELATE uses the CORRELATION function for each pair of variates.

Action with RESTRICT

The VARIATES identifiers may be restricted. If they are restricted they must be restricted in the same way

References

None.

Procedures Used

None.

Similar Procedures

FCORRELATION forms the correlation matrix for a list of variates.

Example

```
CAPTION   'BICORRELATE example' ; STYLE=meta
READ      x[1...5]
  490  450  399  415  441
  461  465  436  426  413
  537  535  448  439  445
  510  522  421  441  444
    *  491  493  516  554
    *  504  455  448  515
    *  418  345  420  463
    *  342  367  437  431
  495  440  514  359  400
  382  400  407  373  358
  376  470  479  525  542
  413  395  423  395  429
  427  433  382  431  381
  481    *  462  485  469
  461    *  496  433  454
  422  492  449  529  495
  405    *  315  364  388
  394    *  438  422  427   :
BICORRELATE [PRINT=correlations, nobservations, test] x[1...5]
FCORRELATION [PRINT=correlations, test] x[1...5]
```

BIOMETRIS

P.W. Goedhart

Accesses information, examples and source of the Biometris Procedure Library

contents previous next

Options

`PRINT` = *string tokens*	Printed output required (`information`, `allfiles`); default `*`, i.e. no printing, or `information` when no other options or parameters have been set
`CONTENTS` = *text*	Saves the contents of the Biometris Procedure Library
`LIBRARYFILE` = *text*	Saves the full filename of the Biometris Procedure Library
`EXAMPLESFILE` = *text*	Saves the full filename of the backingstore file in which the example and source code of all Biometris procedures is stored
`WINBATCHEXECUTABLE` = *text*	Saves the full filename of the external WinBatch executable used by some of the Biometris procedures; only useful for the Windows implementation of GenStat
`DOTNETEXECUTABLE` = *text*	Saves the full filename of the external C# executable used by procedures which employ the `SUSPEND` mechanism
`DEVICE` = *scalar*	Saves the default device used by the `DDEVICE` procedure; default 4 for the Windows implementation
`EXIT` = *scalar*	Saves a scalar which is used internally by other Biometris procedures
`GENDIRECTORY` = *text*	Saves the main GenStat directory

Parameters

`PROCEDURE` = *texts*	Single-valued texts indicating the procedures about which the information is required
`EXAMPLE` = *texts*	To store the example for each procedure
`SOURCE` = *texts*	To store the source code for each procedure
`DATA` = *texts*	To store example data, if applicable, for each procedure

Description

Procedure `BIOMETRIS` allows you to obtain an example of the use of any procedure in the Biometris Procedure Library, also to access the source code of any procedure, so that you can see how it works, or modify it. For procedures which employ the `PASS` mechanism, Fortran code of subroutines is also available. The names of procedures for which examples and source code are required should be listed, in quotes, using the `PROCEDURE` parameter. The `EXAMPLE` parameter can be used to specify the identifier of a text to store each example and the `SOURCE` parameter to store the source code. The `DATA` parameter stores example data used in the example program for some procedures. The following code would run an example of the `RSELECT` procedure.

```
BIOMETRIS 'RSELECT' ; EXAMPLE=ExRselect
EXECUTE   ExRselect
```

The `PRINT=information` setting prints a list of index lines giving brief details about the Biometris procedures. It also prints the full name of all the files which are relevant for the Biometris Procedure Library: (1) the Procedure Library file, (2) the backingstore file with the examples and source code, (3) the full filename of the external WinBatch executable file which is used by some procedures and (4) the full filename of the C# executable used by procedures which employ the `SUSPEND` mechanism. These can also be stored by setting options `LIBRARYFILE`, `EXAMPLESFILE`, `WINBATCHEXECUTABLE` and `DOTNETEXECUTABLE` respectively. The `GENDIRECTORY` option saves the main GenStat directory.

The `PRINT=allfiles` setting outputs the example, source code and associated Fortran code of all Biometris procedures in a directory named "BiometrisSource" below the current working directory.

The `CONTENTS` option can be used to save the contents of the Biometris Procedure Library. The `DEVICE` option saves the default device which is used by the `DDEVICE` procedure. The default value is 4.

Its main use is in the DDEVICE procedure. The EXIT option saves a scalar which is used internally by other Biometris procedures.

Method

The examples, source code and example data are held in a backing-store file. These are accessed using standard retrieval of text structures.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

None.

Similar Procedures

LIBEXAMPLE accesses examples and source code of library procedures in the GenStat Procedure Library.

Example

```
CAPTION   'BIOMETRIS example' ; STYLE=meta
BIOMETRIS [PRINT=information]
BIOMETRIS 'RSELECT' ; EXAMPLE=ExRselect
PRINT     ExRselect ; FIELDWIDTH=1 ; JUSTIFICATION=left ; SKIP=0
EXECUTE   ExRselect
```

CHPOINTER

L.C.P. Keizer & J.T.N.M. Thissen

Checks identifier equivalence of structures in two pointers

contents previous next

Options

PRINT = *string token*	What to print (information); default *
CASE = *string token*	Whether lower- and upper-case (small and capital) letters are to be regarded as identical in identifiers (significant, ignored); default significant

Parameters

CHECKPOINTER = *pointers*	Pointer whose structures are checked ; must be set
TARGETPOINTER = *pointers*	Pointer whose structures are compared with the structures in CHECKPOINTER; must be set
PRESENT = *variates*	Saves a variate of the same length as the CHECKPOINTER parameter with elements 1 (structure present in TARGETPOINTER) or 0 (structure absent in TARGETPOINTER)
ALLPRESENT = *scalars*	Scalar to save whether all structures of CHECKPOINTER are present in TARGETPOINTER (1) or at least one structure is absent (0)

Description

Procedure CHPOINTER can be used to check whether structures of the CHECKPOINTER parameter are present in the TARGETPOINTER parameter or not. Note that only the first 32 characters of an identifier are relevant. The CASE option specifies whether the case of letters (small and capital) in the identifiers of CHECKPOINTER and TARGETPOINTER should be regarded as significant or ignored when comparing the identifiers. The PRESENT parameter can be used to save a variate of the same length as the CHECKPOINTER parameter with elements 1 and 0, indicating whether the corresponding structure of CHECKPOINTER is present (1) in TARGETPOINTER or absent (0). The ALLPRESENT parameter saves whether all structures of CHECKPOINTER are present in TARGETPOINTER (1) or at least one structure is absent (0).

PRINT=information prints presence of the structures and a warning when structures differ only in lower- and upper-case of their letters.

Method

CHPOINTER uses standard GenStat directives for data manipulation.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION   'CHPOINTER example' ; STYLE=meta
POINTER   [VALUES=AA,b,C,d] check
POINTER   [VALUES=aa,e] target
CHPOINTER [PRINT=information ; CASE=ignored] CHECKPOINTER=check ; \
          TARGETPOINTER=target
POINTER   [VALUES=AA,aa] case
CHPOINTER CHECKPOINTER=case ; TARGETPOINTER=target ; PRESENT=present
PRINT     case, present
```

CHSTRUCTURE

J.T.N.M. Thissen & L.C.P. Keizer

Checks attributes of structures

Options

PRINT = *string token*	What to print (information); default *

Parameters

STRUCTURES = *pointers*	Pointer to structures to check; must be set
TYPE = *texts*	Saves a text of the same length as the STRUCTURES parameter with the type of the structures
DECLARED = *variates*	Saves a variate of the same length as the STRUCTURES parameter with elements 1 if the structure is declared or 0 if the structure is not declared
ALLDECLARED = *scalars*	Scalar to save whether all the elements of the STRUCTURES parameter are declared (1) or at least one structure is not declared (0)
PRESENT = *variates*	Saves a variate of the same length as the STRUCTURES parameter with elements 1 if the structure has values or 0 if the structure has no values
ALLPRESENT = *scalars*	Scalar to save whether all the elements of the STRUCTURES parameter have values (1) or at least one structure has no values (0)
NVALUES = *variates*	Saves a variate of the same length as the STRUCTURES parameter with the number of values of the structures
NMV = *variates*	Saves a variate of the same length as the STRUCTURES parameter with the number of missing values of the structures

Description

Procedure CHSTRUCTURE can be used to check structures with respect to declaration, presence of values and the number of (missing) values. The STRUCTURES parameter defines the identifiers to check. The TYPE, DECLARED, PRESENT, NVALUES and NMV parameters all have the same length as the STRUCTURES pointer. The TYPE text saves an asterisk (*) for non-declared identifiers. The DECLARED parameter saves a 1 for declared structures and a 0 otherwise. The PRESENT parameter saves a 1 for structures with (possibly missing) values and a 0 otherwise. The parameters NVALUES and NMV save the number of values and the number of missing values respectively. The ALLDECLARED parameter saves whether all structures of the STRUCTURES parameter are declared (1) or at least one is not (0), whereas the ALLPRESENT parameter saves whether all structures of the STRUCTURES parameter have values (1) or at least one has not (0).

The option setting PRINT=information prints an overview of the different attributes of the structures.

Method

CHSTRUCTURE uses standard GenStat directives for data manipulation.

Action with RESTRICT

Restrictions on structures in the STRUCTURES pointer are ignored, i.e. the number of (missing) values is calculated for unrestricted vectors.

References

None**.**

Procedures Used

None**.**

Similar Procedures

None.

Example

```
CAPTION     'CHSTRUCTURE example' ; STYLE=meta
SCALAR      scal
VARIATE     [NVALUES=10] vari1
VARIATE     [VALUES=5(*)] vari2
VARIATE     [VALUES=1...3] vari3
TEXT        text
SYMMETRIC   [ROWS=10] symm
POINTER     [VALUES=scal, vari1, vari2, vari3, text, symm, not1, not2] input
CHSTRUCTURE [PRINT=information] input ; TYPE=type ; DECLARED=declared
PRINT       input, type, declared
```

COMPACT

P.W. Goedhart

Compacts numerical vectors by storing unique rows and the number of times these occur

contents previous next

Options

`WEIGHTS` = *variate*	Saves the weights, i.e. the number of times each row occurs in the original data
`EXPAND` = *factor*	Saves a factor that can be used to re-generate the original data
`NEWVECTORS` = *pointer*	Saves the new compacted vectors; if `NEWVECTORS` is not set the compacted values are saved in the `VECTORS` parameter

Parameters

`VECTORS` = *variates* and/or *factors*	Numerical vectors that must be compacted

Description

Procedure `COMPACT` can be used to generate a new dataset which has unique rows and to store the number of times each row occurs in the original dataset. This can for instance be used to speed up fitting of a generalized linear model, such as implemented in `RBETABINOMIAL` or `RLOGITNORMAL`, considerably. The `VECTORS` parameter specifies the numerical vectors that must be compacted. The new vectors can be saved by means of the `NEWVECTORS` options; if this is not set, the values of the `VECTORS` are replaced by the compacted structures. The `WEIGHTS` option can be used to save the number of times each row occurs in the original dataset. The `EXPAND` options can be used to save a factor which can be used to re-generate the original data; see the example program

Method

First factors are formed from each variate and then the ordinal levels of all factors (formed and specified) are used to create a new factor which classifies the rows. The `TABULATE` directive is then used to compact the vectors.

Action with RESTRICT

Restrictions are not allowed.

References

None**.**

Procedures Used

None**.**

Similar Procedures

None.

Example

```
CAPTION   'COMPACT example' ; STYLE=meta
VARIATE   [VALUES=1,2,3, 1,2,3, 1,2,3, 1] variate ; DECIMALS=1
FACTOR    [LABELS=!t(A, BB) ; VALUES=6(1), 4(2)] factor
COMPACT   [WEIGHT=weight ; EXPAND=expand ; NEWVECTORS=new] variate, factor
PRINT     expand, variate, factor ; FIELD=10,2(26)
PRINT     weight, new[] ; FIELD=10,2(26)
PRINT     expand, new[]$[expand] ; FIELD=10,2(26)
```

DBBIPLOT

J.T.N.M. Thissen

Produces a high-resolution graphical biplot

contents previous next

Options

`XUPPER` = *scalar*	Upper bound for x- and y-axis in the individuals plot
`VXUPPER` = *scalar*	Upper bound for x- and y-axis in the variates plot
`XMARKS` = *scalar* or *variate*	Distance between each tick mark on x- and y-axis (scalar) or positions of the marks in the individuals plot
`VXMARKS` = *scalar* or *variate*	Distance between each tick mark on x- and y-axis (scalar) or positions of the marks in the variates plot
`XTITLE` = *text*	Title for the x-axis in the individuals plot
`VXTITLE` = *text*	Title for the x-axis in the variates plot
`YTITLE` = *text*	Title for the y-axis in the individuals plot
`VYTITLE` = *text*	Title for the y-axis in the variates plot
`LABELS` = *text*	Labels at each point in the individuals plot
`VLABELS` = *text*	Labels at each point in the variates plot
`SYMBOLS` = *scalar, pointer, factor* or *text*	Plotting symbols: scalar for special symbols, pointer for user defined symbols, text or factor for character symbols
`VSYMBOLS` = *string token*	What to draw at the end of the line (`arrowhead`, `line`, `none`); default `arrowhead`
`COLOUR` = *scalar*	Number of the colour used in the individuals plot
`VCOLOUR` = *scalar*	Number of the colour used in the variates plot
`VLINESTYLE` = *scalar*	Style for the lines in the variates plot
`SCREEN` = *string token*	Whether to clear the screen before plotting the individuals plot or to continue plotting on the old screen (`clear`, `keep`); default `clear`
`VSCREEN` = *string token*	Whether to clear the screen before plotting the variates plot or to continue plotting on the old screen (`clear`, `keep`); default `clear`

Parameters

`COORDINATES` = *matrices*	Scores for the individuals
`VCOORDINATES` = *matrices*	Scores for the variates

Description

Procedure `DBBIPLOT` produces a high-resolution graphical biplot either by employing the saved results of the `BIPLOT` procedure or by specifying explicitly the matrices of scores for individuals and for variates. Gabriel (1971) provides a full description of the technique. The scores for the individuals, contained in a matrix, must be specified by the `COORDINATES` parameter and the scores for the variates, also contained in a matrix, must be specified by the `VCOORDINATES` parameter. Although both matrices can have any dimension, only the first two columns are used. The options can be used to change the appearance of the graph. Option names starting with a `V` relate to the variates plot and the other options relate to the individuals plot.

The individuals plot is just a graph with the scores for individuals represented by dots (default) and labelled by numbers 1 to n (default). The dots are drawn with pen 1 and the labels with pen 2. Options `SYMBOLS` and `LABELS` can be used to change these default settings. If no labels are required the `LABELS` text structure should contain strings with spaces only.

The variates plot gives lines from each point to the origin. Option `VSYMBOLS` specifies what must be drawn at the end of the line. Option `VLABELS` can be used to label the lines. By default the letters of the alphabet are used. The lines are drawn with pen 3, the labels with pen 4 and the arrow-head or perpendicular line with pen 5. The other options are self explanatory.

It is not necessary to specify both matrices. This gives for instance the opportunity to extend biplots to triplots (Gower and Hand, 1996) by using DBBIPLOT twice. In that case the second biplot should have the option setting SCREEN=keep or VSCREEN=keep.

Method

DBBIPLOT calculates the positions of the labels alongside the points for the individuals and the endpoints of the lines for the variates. Then points for the individuals and/or lines for the variates are plotted in the same graph.

Action with RESTRICT

Not relevant.

References

Gabriel, K.R. (1971). The biplot graphic display of matrices with application to principal component analysis. *Biometrika*, **58**, 453-467.

Gower, J.C. and Hand, D.J. (1996). *Biplots.* Chapman & Hall, London.

Procedures Used

FTEXT.

Similar Procedures

Procedure BIPLOT calculates the COORDINATES and VCOORDINATES matrices for producing a biplot. Procedure DBIPLOT produces a similar graph.

Example

```
CAPTION   'DBBIPLOT example', 'Data taken from the BIPLOT example', ' ' ; \
          STYLE=meta, 2(plain)
VARIATE   [NVALUES=20] v[1...7]
READ      v[]
  4  11  4  28  31  17  21      5  11  5  29  30  16  21
  7   9  6  25  30  17  23      3   9  5  28  32  12  15
  5  15  6  29  34  18  21      3  10  5  23  27  17  20
  3  10  7  24  28  18  21      3  13  7  29  34  18  21
  3  10  5  26  21  17  28      5  10  6  26  30  16  23
  7   9  5  26  30  16  23      4  11  8  27  31  17  22
  3  12  6  26  31  18  24      4  11  7  26  31  18  23
  6  10  9  28  31  21  27      4  12  9  27  32  16  25
  5  12  8  29  33  15  22      4  14  6  23  29  16  19
  4  10  6  25  29  19  22      3  15  7  25  29  16  19 :
TEXT      [VALUES=va,vb,vc,vd,ve,vf,vg] vlabs
BIPLOT    [METHOD=var ; PRINT=singular,scores ; VLABELS=vlabs] v ; \
          COORDINATES=comat ; VCOORDINATES=vcomat
TEXT      title ; 'Example of DBBIPLOT: AXIS-1 variates'
DBBIPLOT  [VLABELS=vlabs ; VXTITLE=title] COORDINAT=comat ; VCOORDINAT=vcomat
```

DDEVICE

P.W. Goedhart

Selects a graphics device and opens a corresponding graphics file

contents previous next

Options

MESSAGE = *string token*	Whether to print a one-line message with the device number and the corresponding graphics file (yes, no); default no

Parameters

NUMBER = *scalars*	Device number; the default value is *, i.e. 4 for the Windows implementation of GenStat and 6 for other implementations
NAME = *texts*	External name of the graphics file; default * uses the name of the current input file. When there is no current input file the default name is "genstat"; the default extension is 100 * NUMBER + 1
ENDACTION = *string token*	Action to be taken after completing each plot (continue, pause)

Description

Directive DEVICE switches between (high-resolution) graphics devices. If a file-based device is selected, the OPEN directive has to be used to open a file to receive the graphical output, e.g.

```
OPEN    'plot.401' ; CHANNEL=4 ; FILETYPE=graphics
DEVICE  4
```

Procedure DDEVICE combines these two statements with sensible default settings.

The NUMBER parameter selects the graphics device number. If the NUMBER parameter is unset, the procedure switches to device 4 for the Windows implementation of GenStat and to device 6 for other implementations.

The NAME parameter specifies the filename of the graphical output file. If the NAME parameter is unset, the name of the graphics file is identical to the name of the current input file. When there is no current input file, the default name of the graphics file is "genstat". The default extension of the graphics file is 100 * NUMBER + 1. This extension can be useful if graphical output is send to individual graphics files in which case the extension is automatically incremented for each new plot. If there is already a graphics file attached to the specified device, that graphics file is used and a warning message is printed. Note that no graphical output file is opened when device number 1 is selected, because graphical output is then written to screen.

The ENDACTION parameter controls the action taken by default at the end of the plot. The MESSAGE option can be used to print a one-line message with the device number and the name of the graphics file which is opened on the corresponding graphics channel.

Method

The ENQUIRE directive is used to obtain the name of the current input file.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

The BIOMETRIS procedure is used to determine the implementation of GenStat.

Similar Procedures

Procedure SETDEVICE opens a graphical file and specifies the device number on basis of its extension.

Example

```
CAPTION   'DDEVICE example' ; STYLE=meta
DDEVICE   [MESSAGE=yes]
PRINT     'Restore default graphics device' ; SKIP=0
CLOSE     CHANNEL=4,6 ; FILETYPE=graphics
DEVICE    1
```

DIRLIST

P.W. Goedhart & L.C.P. Keizer

Provides details about (wild-carded) files in a specified directory

contents previous next

Options

`PRINT` = *string token*	What to print (`filelist`); default `filelist`
`DIRECTORY` = *text*	Single-valued text which specifies the directory for the file list; default `*`, i.e. the current working directory
`SAVEDIRECTORY` = *text*	Saves the full name of `DIRECTORY`
`EXISTDIRECTORY` = *scalar*	Saves whether `DIRECTORY` exists (1) or not (0)
`SUBDIRECTORIES` = *text*	Saves the subdirectories of the specified directory
`CASE` = *string token*	Case to use for letters of `SAVEDIRECTORY`, `NAME`, `SURNAME` and `EXTENSION` (`given`, `lower`, `upper`, `changed`, `title`); default `given` leaves the case of each letter unchanged
`SINDEX` = *string tokens*	Defines the ordering of the printed filelist and of the saved parameters (`name`, `surname`, `extension`, `size`, `date`, `time`, `attribute`); default `name`
`SDIRECTION` = *string token*	Order in which to sort (`ascending`, `descending`); default `ascending`
`NOMESSAGE` = *string tokens*	Which warning messages to suppress (`readdeny`, `existdirectory`, `nofilesfound`); default `*`

Parameters

`FILES` = *texts*	Files for file list, may contain wildcards `*` and `?`, must not contain drives or directories
`PRESENT` = *variates*	Saves the number of files in each line of `FILES`; 0 indicates that the corresponding file is not present and `*` that the directory does not exist
`NAME` = *texts*	Saves the name of the files
`SURNAME` = *texts*	Saves the surname of the files, i.e. the name excluding the period and extension
`EXTENSION` = *texts*	Saves the extension of the files, excluding the leading period
`SIZE` = *variates*	Saves the size of the files
`DATE` = *texts*	Saves the date of the files
`TIME` = *texts*	Saves the time of the files
`ATTRIBUTE` = *texts*	Saves the attributes of the files; this is a 4 letter string with successively r(eadonly, a(rchive), s(ystem), h(idden). A hyphen in any of these positions indicates that the specified attribute is off.

Description

Procedure `DIRLIST` can be used to obtain information about files in a specified directory. It can also be used to obtain a wild-carded directory list. The directory can be specified by the `DIRECTORY` option; default is to look for files in the current working directory. Note that the double backslash (\\) is required in `DIRECTORY` because in GenStat a single "\" is treated as indicating a continuation on the next line. However, you can use a single "/" instead of de double backslash (\\).

The files for which information is required must be specified by the `FILES` parameter. The number of files present can be saved by the `PRESENT` parameter; this parameter is of the same length as `FILES`. The `PRESENT` parameter is filled with missing values when `DIRECTORY` does not exist. Further information about the files can be saved by means of parameters `NAME`, `SURNAME`, `EXTENSION`, `SIZE`, `DATE`, `TIME` and `ATTRIBUTE`. These structures are all of length `SUM(PRESENT)`, the length depends on whether files exist and whether wildcards are used. In case a file is opened by another program in read deny mode, its size cannot be determined and the size is set to missing. The full path of `DIRECTORY` can be saved by means of the SAVEDIRECTORY option. Option EXISTDIRECTORY saves whether the DIRECTORY exists, and the SUBDIRECTORIES option can be used to save the subdirectories of the specified directory.

The CASE option can be used to change the case of the saved text structures NAME, SURNAME, EXTENSION and SAVEDIRECTORY. The title setting of CASE changes the case of all letters to lowercase, except the first letter which is changed to uppercase. The SINDEX option defines the ordering of the printed file list and of the saved parameters. The DIRECTION option controls whether the ordering is into ascending or descending order. The default settings are SINDEX=name and SDIRECTION=ascending.

The PRINT option controls printed output. The NOMESSAGE option can be used to suppress warning messages in case the SDIRECTORY does not exist, or when no FILES are found or when one or more files are in read deny mode, i.e. when files are in use by another program. For files in read deny mode, the size and time are set to missing, while the date is set to the date at which DIRLIST is called.

Method

The SUSPEND [CONTINUE=no] directive is used to invoke an external WinBatch executable.

Action with RESTRICT

Restrictions on the DIRECTORY option and the FILES parameter are ignored.

References

None.

Procedures Used

The BIOMETRIS procedure is used to retrieve the filename of the external WinBatch executable.

Similar Procedures

QDIRECTORY returns a directory selected by means of a directory browse dialog box on screen. QFILENAME returns a single filename selected by means of a file open box on screen.

Example

```
CAPTION   'DIRLIST example' ; STYLE=meta
DIRLIST   [SAVEDIRECTORY=savedir ; CASE=title] FILES=!t('*.*')
PRINT     savedir
TEXT      files ; !t('*.ini', '*.hlp')
DIRLIST   [PRINT=* ; DIRECTORY='C:/WINDOWS' ; EXISTDIRECTORY=exist] \
          FILES=files ; PRESENT=present ; NAME=name ; SURNAME=surname ; \
          EXTENSION=extension
PRINT     files, present ; FIELD=14
IF exist .AND. SUM(present)
  PRINT   name, surname, extension ; SKIP=3
ENDIF
```

DORDINAL

J.T.N.M. Thissen

Plots and displays the results of a simple ordinal logistic regression model

contents previous next

Options

PRINT = *string tokens*	Output required (plot, curve, line, predictions, pairtest, items); default plot, curve, line, predictions, pairtest
CTITLE = *text*	General title for the display of curves; default *
LTITLE = *text*	General title for the display of lines; default *
CYTITLE = *text*	Title for the y-axis in the display of curves; default *
LYTITLE = *text*	Title for the y-axis in the display of lines; default *
CXTITLE = *text*	Title for the x-axis in the display of curves; default *
LXTITLE = *text*	Title for the x-axis in the display of lines; default *
SCTITLE = *scalar*	Multiplier used in the calculation of the size in which to draw CTITLE; default 1
SLTITLE = *scalar*	Multiplier used in the calculation of the size in which to draw LTITLE; default 1
SCYTITLE = *scalar*	Multiplier used in the calculation of the size in which to draw CYTITLE; default 1
SLYTITLE = *scalar*	Multiplier used in the calculation of the size in which to draw LYTITLE; default 1
SCXTITLE = *scalar*	Multiplier used in the calculation of the size in which to draw CXTITLE; default 1
SLXTITLE = *scalar*	Multiplier used in the calculation of the size in which to draw LXTITLE; default 1
SORT = *string token*	Whether to sort the means in the diagram in ascending order (yes, no); default yes
XLGAP = *scalar*	Gap on the x-axis for the labels of the treatment factor; default 0.02
NPAGES = *scalar*	Number of pages for plotting the line plot

Parameters

TREATMENTSTRUCTURE = *factor*	Defines the treatment factor of the model
BLOCKSTRUCTURE = *factor*	Defines the block factor of the model
TPROBABILITIES = *symmetric matrix*	Saves the t-probabilities of tests of pairwise comparisons
PREDPERCENTAGES = *pointer*	Saves the predictions in the categories expressed as percentages

Description

Ordinal logistic regression can be performed by using the MODEL directive with option settings YRELATION=cumulative, DISTRIBUTION=multinomial and LINK=logit. This model is also called the proportional-odds model, see McCullagh & Nelder (1989). Procedure DORDINAL can be used to aid in the interpretation of the results of a simple ordinal logistic regression model, i.e. a model with only one treatment factor and possibly one block factor.

A call to DORDINAL must be preceded by an appropriate MODEL statement, a TERMS [FULL=yes] statement and a FIT statement. The TREATMENTSTRUCTURE and BLOCKSTRUCTURE parameters of DORDINAL should be set to the factors specified with TERMS and FIT. Plotting and printing of the results is controlled by the PRINT option with the settings: plot to display the fitted logistic distributions over the categories for each level of the treatment factor in a high-resolution plot; curve to display the distributions as a curve and line to display the distributions as a line between the 2.5% and 97.5% point of the logistic distributions (default both plots are produced unless the number of levels of the TREATMENTSTRUCTURE factor is greater than 20, in which case only the line plot is drawn); predictions to print the predicted percentages of the numbers of observations in each category; and pairtest to perform t-tests for all pairwise differences between the levels of the treatment factor. If pairtest is specified procedure PPAIR with option PRINT=groups is used to print the diagram of significant differences; the items setting prints significant differences in another format (see procedure PPAIR). If the labels of the treatment factor are too long to fit in the default left hand side of the plot, the XLGAP option can be used to widen that gap. With the NPAGES option the number of pages can be supplied if the number of treatments is too big for a line plot on one page.

The t-probabilities of the tests of pairwise comparisons can be saved by parameter TPROBABILITIES. The SORT option controls whether the means in the diagram of PPAIR are sorted into ascending order. The predictions in the categories, expressed as percentages, can be saved by the PREDPERCENTAGES parameter. This parameter is a pointer (with length the number of categories) referring to variates with length the number of levels of the treatment.

All other options relate to the graphical environment of the plot. The CTITLE option can be used to provide a title for the graph of the curves and LTITLE for the graph of the lines. Titles can be added to the axes using the CYTITLE, LYTITLE, CXTITLE and LXTITLE options. By default the names of the category structures are plotted alongside the y-axis and the labels (or levels) of the treatment factor alongside the x-axis. The pensizes for the titles can be changed by using the options SCTITLE, SLTITLE, SCYTITLE, SLYTITLE, SCXTITLE and SLXTITLE.

Method

Procedures PAIRTEST and PPAIR are used to test all pairwise comparisons between the levels of the TREATMENTSTRUCTURE factor. The saved fitted values are used to calculate the predictions, and the formula of the logistic distribution is used to plot the logistic curves and to display the line plot.

Action with RESTRICT

Not relevant. The parameters TREATMENTSTRUCTURE and BLOCKSTRUCTURE are only needed to distinguish between the treatment and block factor.

References

McCullagh, P. and Nelder, J.A. (1989). *Generalized linear models (second edition)*. Chapman and Hall, London.

Procedures Used

CHECKARGUMENT, PAIRTEST, PPAIR, FFRAME, FTEXT.

Similar Procedures

None.

Example

```
CAPTION   'DORDINAL example' ; STYLE=meta
FACTOR    [LABELS=!t(T1, T2, T3, T4, T5, T6, T7) ; VALUES=8(1...7)] Treatm
FACTOR    [LEVELS=8 ; VALUES=(1...8)7] Block
READ      Healthy, Light, Middle, Heavy
  477 115  38  20    413 231  43   0    372 136  67  20    417 135  45   0
  449 116   6   0    409 147 142   0    387 149  71   0    354 201  21   0
   82 344 141  52    107 279 187  28     73 340 157  54     43 384 232   0
   95 286 173  32    100 372 144   0    149 239  85   0     61 424 131   0
   55 299 206  57     25 307 245  71    120 182 239  21     51 356 146 114
   77 388 117   0    160 301  37   0     71 406  58  27    168 289 109   0
  246 352  93   0    128 360 173  16    198 318  99  40    163 362 117  47
  263 239 113   8    269 249  71   0    181 296 102   0    219 345  77   0
  226 385  80   6    231 306  93   0    284 362  38   0    216 434  31   0
  316 203  54   0    270 333  72   0    288 253  37   0    229 308 128   0
  180 351 105  64    129 437  60  66    124 423  81  20    194 341 103  18
  164 364  65   0    180 334 113   0    269 251  69   0    291 232  37   0
  159 404 104  32    227 409  42   0    227 357  50  33    253 377  57   0
  287 243  53  16    300 298  30   0    430 122  22   0    296 196  70   0  :
MODEL     [DISTRIBUTION=multinomial ; LINK=logit ; YRELATION=cumulative ; \
          DISPERSION=*] Healthy, Light, Middle, Heavy
TERMS     [FULL=yes] Treatm + Block
FIT       Block + Treatm
DORDINAL  [CTITLE='DORDINAL' ; LTITLE='DORDINAL'] Treatm ; Block
DORDINAL  [PRINT=pairtest ; SORT=no] Treatm ; Block
```

EDCONTINUOUS

M.J.W. Jansen, J.C.M. Withagen & J.T.N.M. Thissen

Calculates equivalent deviates for continuous distributions

contents previous next

Options

DISTRIBUTION = *string token*	Type of distribution required (beta, gamma, lognormal, normal, uniform); default normal
METHOD = *string token*	Method by which the defining parameters of the distribution are specified (moments, quantiles); default moments
MEAN = *scalar*	Mean of distribution; default *
VARIANCE = *scalar*	Variance of distribution; default *
PROPORTIONS = *variate*	Two cumulative lower probabilities of distribution; default *
QUANTILES = *variate*	Two quantiles (equivalent deviates) corresponding to PROPORTIONS; default *
LOWER = *scalar*	Lower bound of beta, gamma, lognormal or uniform distribution; default 0
UPPER = *scalar*	Upper bound of beta or uniform distribution; default 1

Parameters

CUMPROBABILITY = *variates* or *scalars*	Cumulative lower probabilities for which equivalent deviates are required; must be set
DEVIATE = *variates* or *scalars*	To save equivalent deviates corresponding to CUMPROBABILITY

Description

Procedure EDCONTINUOUS calculates equivalent deviates corresponding to given cumulative lower probabilities for five continuous distributions: beta, gamma, lognormal, normal and uniform. The CUMPROBABILITY parameter specifies the cumulative lower probabilities and the corresponding equivalent deviates are saved by means of the DEVIATE parameter. The DISTRIBUTION option specifies the type of distribution. The METHOD option specifies how the parameters of the distribution are defined. When METHOD=moments the first two moments must be set by the MEAN and VARIANCE options. Alternatively, when METHOD=quantiles the distribution is characterised by a pair of cumulative lower probabilities with corresponding quantiles, and options PROPORTIONS and QUANTILES must be set. The uniform distribution is characterised by the LOWER and UPPER option settings, and other options are ignored. Lower and upper bounds for the other distributions can be specified by options UPPER and LOWER; these must be compatible with other option settings.

Method

Internal calls are made to Genstat's ED-functions EDNORMAL, EDBETA and EDGAMMA. In most cases, the required ED-function parameters are derived from simple, well-known relations between ED-function parameters and moments or quantiles. However, when a beta or gamma distribution is specified by two quantiles, the ED-function parameters are derived by means of the FITNONLINEAR directive, which may cause numerical problems.

Action with RESTRICT

Deviates are only calculated for the set of units to which CUMPROBABILITY is restricted. Other units will remain unaffected.

References

None.

Procedures Used

None.

Similar procedures

`GRANDOM` generates pseudo-random numbers from probability distributions. `GMULTIVARIATE` generates pseudo-random numbers from multivariate normal or Student's t distribution. `GRMULTINORMAL` generates pseudo-random numbers from the multivariate normal distribution

Example

```
CAPTION      'EDCONTINUOUS example' ; STYLE=meta
VARIATE      cum ; !(0.01, 0.02 ... 0.99)
EDCONTINUOUS [DIST=normal ; METHOD=quantiles ; PROPORTION=!(.05, .95) ; \
             QUANTILES=!(6.9, 8.2)] CUMPROBABILITY=cum ; DEVIATE=v[1]
EDCONTINUOUS [DIST=beta ; METHOD=quantiles ; PROPORTION=!(.25, .75) ; \
             QUANTILES=!(0.3, 0.5)] CUMPROBABILITY=cum ; DEVIATE=v[2]
EDCONTINUOUS [DIST=gamma ; MEAN=2 ; VARIANCE=1] CUMPROBABILITY=cum ; \
             DEVIATE=v[3]
TEXT         title ; 'Example of EDCONTINUOUS: v[1]'
DHISTOGRAM   [WINDOW=5 ; KEY=0 ; TITLE=title    ; SCREEN=keep] v[1]
DHISTOGRAM   [WINDOW=6 ; KEY=0 ; TITLE='v[2]'   ; SCREEN=keep] v[2]
DHISTOGRAM   [WINDOW=7 ; KEY=0 ; TITLE='v[3]'   ; SCREEN=keep] v[3]
DGRAPH       [WINDOW=8 ; KEY=0 ; TITLE='v[2,3]' ; SCREEN=keep] v[2] ; v[3]
```

EMMULTINORMAL

P.W. Goedhart

Estimates parameters of the multivariate normal distribution for data with missing values

contents previous next

Options

`PRINT` = *string tokens*	Printed output required (`pattern`, `monitoring`, `estimates`); default `pattern`, `estimates`
`MAXCYCLE` = *scalar*	Maximum number of iterations for EM algorithm; default 200
`TOLERANCE` = *scalar*	Convergence criterion; default 1.0e-4

Parameters

`DATA` = *pointers*	Pointer of variates with observed values
`MEANS` = *variates*	To save the estimate mean vector of the multivariate normal distribution
`VCOVARIANCE` = *symmetric matrices*	To save the estimate variance-covariance matrix of the multivariate normal distribution
`EXPECTED` = *pointers*	To save the expected values for missing data

Description

Procedure `EMMULTINORMAL` estimates the parameters of the multivariate normal distribution for data with missing values using the EM algorithm. The `DATA` pointer specifies the variates for which parameters must be estimated. The estimates can be saved by means of the `MEANS` and `VCOVARIANCE` parameters. Missing values are replaced by conditional expected values in the EM algorithm and these can be saved by means of the `EXPECTED` parameter.

The EM algorithm is an iterative procedure and the maximum number of cycles and the convergence criterion can be set by `MAXCYCLE` and `TOLERANCE`. The `PRINT` options can be set to `pattern` to display the number of observations for each pattern of missing and non-missing values, to `monitoring` for convergence monitoring of the EM algorithm and to `estimates` to print the estimated parameters.

The algorithm can be very slow when there are many variates and/or many units.

Method

Each iteration in the EM algorithm consists of two steps. The E-step of the algorithm computes the expected value of the sufficient statistics conditional on the observed values and the current parameter estimates. The M-step then uses the expected complete-data sufficient statistics to compute an update of the parameter estimates. This is iterated until the maximum of the relative difference of two subsequent estimates of the mean vector is less than the `TOLERANCE` convergence criterion. The algorithm starts with raw means, variances and covariances. Gelman et al (1995) gives a concise description of the algorithm for the multivariate normal distribution. To speed up the algorithm, the E-step is performed for all units with the same pattern of missing and non-missing values instead of for each unit separately.

Action with RESTRICT

Restrictions on variates in the `DATA` pointer are not allowed.

References

Gelman, A., Carlin, J.B., Stern, H.S. & Rubin, D.B. (1995). *Bayesian Data Analysis*. Chapman and Hall. London.

Procedures Used

None.

Similar procedures

None.

Example

```
CAPTION       'EMMULTINORMAL example' ; STYLE=meta
VARIATE       [VALUES=1...4] mean
SYMMETRIC     [ROWS=4 ; VALUES=1, 0.8,1, 0.7,0.6,1, 0.5,0.4,0.3,1] vcov
GRMULTINORMAL [NVALUES=100 ; SEED=32892 ; MEAN=mean ; VCOV=vcov] data
CALCULATE     data[] = MVINSERT(data[] ; URAND(4(0);100).LT.0.4)
EMMULTINORMAL data
```

FDERIVATIVE

P.W. Goedhart

Calculates numerical derivatives of a function

contents previous next

Options

`CALCULATION` = *expressions*	Expression structures to calculate the target function
`PARAMETERS` = *scalars*	Identifiers of the scalars, used by `CALCULATION`, which are arguments of the function
`FUNCTION` = *scalar*	Identifier of the scalar, calculated by `CALCULATION`, which defines the function value
`DATA` = *pointer*	Data to be used within the `CALCULATION` expressions or within the `%FDERIVATIVE` procedure
`METHOD` = *string token*	Numerical method to use (`simple`, `ridders`); default `simple`
`STEPMETHOD` = *string token*	Whether the steplength is absolute or relative with respect to the parameters (`absolute`, `relative`); default `relative`
`STEPSIZE` = *scalar*	Steplength to use in case the `STEPLENGTH` parameter is not set; default `*` uses 0.001 for the simple method and 0.1 for Ridders method
`STEPTOLERANCE` = *scalar*	Smallest absolute value for which a relative steplength must be used. Only relevant for `STEPMETHOD=relative`; default 1.0
`MAXCYCLE` = *scalar*	Maximum number of iterations for Ridders method; default 20
`TOLERANCE` = *scalar*	Error criterion for exiting Ridders iterations; default 0 exits the iterative procedure when the global convergence criterion is satisfied

Parameters

`VALUES` = *pointers*	Values for which the derivative of the function must be calculated; the pointer can be set to scalars, variates or unknown structures
`STEPLENGTH` = *pointers*	Steplengths to use for calculation of each derivative; default * uses the value as set by the `STEPSIZE` option
`D1` = *pointers*	Saves the first order derivatives to the parameters in a pointer to variates
`D2` = *pointers*	Saves the second order derivatives to the parameters in a pointer to variates
`DMIXED` = *pointers*	Saves the mixed second order derivatives to the parameters in a pointer to symmetric matrices
`ERROR` = *variates*	Saves an estimate of the error in the derivatives

Description

`FDERIVATIVE` can be used to numerically approximate the first and second order derivatives of a function. The function can be specified either by setting the `CALCULATION` option to a list of GenStat expressions, or by defining a procedure `%FDERIVATIVE` which returns a function value. The `METHOD` option specifies the numerical procedure to be used. The `simple` setting uses $[F(x+h) – F(x-h)]/(2h)$ to calculate the first order derivative, in which x is the value for which the derivative is required and h is a steplength. The second order derivative is then calculated by means of $[F(x+h) + F(x-h) -2F(x)]/(h^2)$. The mixed second order derivative is only calculated when there are two parameters or more. It is calculated as $[F(x+h,y+h) + F(x-h,y-h) - F(x-h,y+h) - F(x+h,y-h)]/(4h^2)$, where x and y are the values for which the mixed derivative is required. The `ridders` setting employs Ridders (1982) algorithm which is a form of Richardson extrapolation. The latter method is more accurate and more time consuming since more function evaluations with increasingly smaller steplengths are used. The steplength h can be either absolute, as specified by the `STEPMETHOD` option, or relative in which case steps of $h*x$ are used. However when Abs(x) is smaller than `STEPTOLERANCE`, the absolute steplength h is used. For `METHOD=ridders` the initial value of the steplength is not crucial although not too small values are to be used. Press et al (2002) advice to set the steplength for Ridders method to a few tenth. For METHOD=simple the value of the steplength is crucial and it is advised to compare the derivatives for various steplengths.

In case the CALCULATION option is set to a list of expressions, the parameters used in these expressions must be specified by means of the PARAMETERS option and the FUNCTION option must be set to the resulting function value. The DATA option can be employed to provide any data structures that are used in the expressions to calculate the value of the function. The derivatives are calculated for values in the VALUES pointer. The length of this pointer should equal the length of the PARAMETERS list, and the pointer may contain variates and scalars. These values are processed row by row, and the derivatives with respect to each parameter are calculated. The VALUES pointer may contain an unknown structure, as specified by *, in which case the associated derivative is not calculated. In that case the corresponding value of the PARAMETERS list is used in the calculations. A general steplength can be specified by means of the STEPSIZE option. Alternatively parameter specific steplengths can be specified by means of the STEPLENGTH parameter, either as scalars or variates. The general steplength is used for scalars with a missing value in the STEPLENGTH parameter.

In case the CALCULATION option is not set, the user should provide a %FDERIVATIVE procedure which should return the function value for a set of parameters. The DATA option can then be employed to provide any data structures that are needed by _ FDERIVATIVE to calculate the value of the function. In this case the PARAMETERS and FUNCTION option settings are not used. Details are given in the Methods section, and the Example section provides an example.

Derivatives can be saved by means of the parameters D1, D2 and DMIXED. The length of the pointers D1 and D2 is the same as the length of the VALUES pointer, and the variates that are hold by these pointers have the same length as the number of rows in the VALUES pointer. The DMIXED pointer on the other hand contains symmetric matrices, one for each row in the VALUES pointer. The diagonal of each symmetric matrix in DMIXED is set to missing.

Ridders method provides an estimate of the error in the derivative. The ERROR parameter can be set to save, in a variate of length 2, the maximum error of all first order derivatives and the maximum of all second order derivatives. The maximum error is set to zero when D1 and/or D2 is unset. Ridders method is based on function values at increasingly smaller steplengths which results in higher order approximations of the derivative. The number of times the steplength is decreased is limited by the MAXCYCLE option. Moreover when the estimated error is smaller than the setting of TOLERANCE the iterative procedure is exited. By default, i.e. with TOLERANCE=0, the iterative procedure is stopped when the higher order approximation becomes worse.

Method

The implementation of Ridders (1982) uses a transcript of the C++ algorithm DFRIDR in Press et al (2002). The skeleton of the _DERIVATIVE procedure, for calculation of more complex functions, is as follows

```
PROCEDURE [PARAMETER=pointer] '%FDERIVATIVE'
OPTION    'FUNCTION', 'DATA' ; TYPE='scalar', 'pointer' ; \
          SET=yes ; DECLARED=yes ; PRESENT=yes
PARAMETER 'PARAMETERS' ; TYPE='scalar' ; SET=yes ; DECLARED=yes ; PRESENT=yes
ENDPROCEDURE
```

This procedure should return the function value by means of the FUNCTION option for parameters as specified by the PARAMETERS option of FDERIVATIVE. The number of structures in the VALUES parameter of FDERIVATIVE should in principle be equal to the number of PARAMETERS used within %FDERIVATIVE.

Action with RESTRICT

Variates in the VALUES and STEPLENGTH pointers should not be restricted.

References

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (2002). *Numerical recipes in C++. The art of scientific computing, 2nd edition*. Cambridge University Press. Cambridge.

Ridders, C.J.F. (1982). *Advances in engineering software*, **4**, 75-76.

Procedures Used

The subdidary procedure `%FDERIVATIVE` is used.

Similar Procedures

None.

Example

```
CAPTION   'FDERIVATIVE example using the CALCULATION option' ; STYLE=meta
SCALAR    x, y, z, b1, b2 ; 3(1), 2, 3
EXPRESSIO calc  ; !e(fun= b1*LNGAMMA(x) + b2*LNGAMMA(y) + z*LNGAMMA(x*y))
CALCULATE init = URAND(419032 ; 1)
CALCULATE valx,valy,valz = GRUNIFORM(3(10) ; 1 ; 5)
FDERIVATI [CALCULATION=calc ; PARAMETERS=x,y,z ; FUNCTION=fun ; \
          DATA=!p(b1,b2)] !p(valx,valy,valz) ; D1=d1 ; D2=d2 ; DMIXED=dmixed
CAPTION   'Compare numerical derivatives with exact values','' ; STYLE=plain
CALCULATE dx1 = b1*DIGAMMA(valx) + valz*valy*DIGAMMA(valx*valy)
CALCULATE dy1 = b2*DIGAMMA(valy) + valz*valx*DIGAMMA(valx*valy)
CALCULATE dz1 = LNGAMMA(valx*valy)
CALCULATE dx2 = b1*TRIGAMMA(valx) + valz*valy*valy*TRIGAMMA(valx*valy)
CALCULATE dy2 = b2*TRIGAMMA(valy) + valz*valx*valx*TRIGAMMA(valx*valy)
CALCULATE dz2 = 0*valx
CALCULATE dxy = valz*DIGAMMA(valx*valy) + valz*valy*valx*TRIGAMMA(valx*valy)
CALCULATE dxz = valy * DIGAMMA(valx*valy)
CALCULATE dyz = valx * DIGAMMA(valx*valy)
VEQUATE   dmixed ; !p(dummy, d3[1], dummy, d3[2,3], dummy)
CALCULATE maxd1[1...3] = MAX(ABS(dx1,dy1,dz1 - d1[]))
CALCULATE maxd2[1...3] = MAX(ABS(dx2,dy2,dz2 - d2[]))
CALCULATE maxd3[1...3] = MAX(ABS(dxy,dxz,dyz - d3[]))
PRINT     !(#maxd1), !(#maxd2), !(#maxd3) ; FIELD=-12 ; DECIMALS=2
PRINT     dx1,dy1,dz1, d1[] ; DECIMALS=5
PRINT     dx2,dy2,dz2, d2[] ; DECIMALS=5
PRINT     dxy,dxz,dyz, d3[] ; DECIMALS=5

CAPTION   'FDERIVATIVE example using the %FDERIVATIVE procedure' ; STYLE=meta
CAPTION   !t('Procedure returns the log-likelihood of a negative binomial',  \
          'regression model for a fixed aggregation parameter.', \
          'The exact derivative with respect to the aggregation parameter', \
          'can also be calculated. Also see RNEGBINOMIAL.', '') ; STYLE=plain
PROCEDURE [PARAMETER=pointer] '%FDERIVATIVE'
  OPTION    'FUNCTION', 'DATA' ; TYPE='scalar', 'pointer' ; \
            SET=yes ; DECLARED=yes ; PRESENT=yes
  PARAMETER 'PARAMETERS' ; TYPE='scalar' ; SET=yes ; DECLARED=yes ; \
            PRESENT=yes
  GET       [SPECIAL=save]
  CALCULATE kk = PARAMETERS[1]
  CALCULATE save['rsave'][1][2]$[9] = kk   "Sets the aggregation parameter"
  FIT       [PRINT=* ; NOMESSAGE=dis,lev,res,ali,mar,ver,df,inf] #DATA[1]
  RKEEP     [PMODEL=pm] FITTED=fit
  DUMMY     yy ; pm['y']
  CALCULATE FUNCTION = SUM(LNGAMMA(yy+kk) - LNGAMMA(kk) + kk*LOG(kk) + \
            yy*LOG(fit+(yy==0)) - (yy+kk)*LOG(fit+kk) - LNGAMMA(yy+1))
ENDPROCEDURE
```

```
FACTOR    [NVALUES=10; LEVELS=2; LABELS=!t('Continuous','Standby')] mode
READ      [PRINT=* ; SETNVALUES=yes] mode,events,time
    1  5  94.320   2  1  15.720   1  5  62.880   1 14 125.760   2  3   5.240
    1 19  31.440   2  1   1.048   2  1   1.048   2  4   2.096   2 22  10.480 :
CALCULATE logtime = LOG(time)
MODEL     [DISTRIBUTION=negativebinomial ; LINK=log ; OFFSET=logtime] events
VARIATE   aggregation, d1exact, d2exact ; !(1, 1.1 ... 2)
DUMMY     yy ; events
FOR [NTIMES=NVALUES(aggregation) ; INDEX=ii
  SCALAR    kk ; aggregation$[ii]
  MODEL     [DIST=negative ; LINK=log ; OFFSET=logtime ; AGGREGATION=kk] yy
  FIT       [PRINT=*] mode
  RKEEP     FITTED=fit
  CALCULATE d1exact$[ii] = SUM( LOG(kk/(fit+kk)) + (fit-yy)/(fit+kk) + \
            DIGAMMA(yy+kk) -  DIGAMMA(kk) )
  CALCULATE d2exact$[ii] = SUM( (1/kk - 1/(fit+kk) - (fit-yy)/(fit+kk)**2 + \
            TRIGAMMA(yy+kk) - TRIGAMMA(kk) ))
ENDFOR

POINTER   [NVALUES=1] data
FORMULA   data[1] ; !f(mode)
FDERIVATI [DATA=data ; METHOD=simple]  !p(aggregation) ; D1=ds1 ; D2=ds2
FDERIVATI [DATA=data ; METHOD=ridders] !p(aggregation) ; D1=dr1 ; D2=dr2
CALCULATE maxds1, maxdr1 = MAX(ABS(d1exact - ds1[],dr1[]))
CALCULATE maxds2, maxdr2 = MAX(ABS(d2exact - ds2[],dr2[]))
PRINT     maxds1, maxdr1, maxds2, maxdr2 ; FIELD=-12 ; DECI=2
PRINT     aggregation, d1exact,ds1[],dr1[], d2exact,ds2[],dr2[] ; DECI=1,6(5)
```

FEXPAND

J.T.N.M. Thissen

Forms a variate and/or factor by expanding a structure a specified number of times

contents previous next

Options

None.

Parameters

STRUCTURE = *identifiers*	Structure (scalar, variate, text, table, matrix, symmetricmatrix, diagonalmatrix) to be expanded
NOBSERVATIONS = *identifiers*	Numerical structure (scalar, variate, table, matrix, symmetricmatrix, diagonalmatrix) specifying the number of times each value of STRUCTURE must be expanded
VARIATE = *variates*	Variate to save the expanded values
FACTOR = *factors*	Factor to save the expanded values

Description

Procedure FEXPAND expands the values of the STRUCTURE parameter a number of times as specified by parameter NOBSERVATIONS. Each value of the STRUCTURE parameter is copied as many times as the corresponding value of the NOBSERVATIONS parameter. The VARIATE and FACTOR parameters can be used to save the expanded structure as a variate and a factor. The STRUCTURE parameter can be set to a scalar, variate, text, table, matrix, symmetricmatrix or diagonalmatrix. If the STRUCTURE parameter is set to a text, output structure VARIATE must not be set. The NOBSERVATIONS parameter can be set to the same type of structures as the STRUCTURE parameter, with the exception of a text structure. The STRUCTURE and NOBSERVATIONS parameters must have the same number of values.

Missing values are not allowed in the NOBSERVATIONS parameter. If NOBSERVATIONS contains a zero the corresponding value of the STRUCTURE parameter is omitted.

Method

The EXPAND function is used to calculate the new variate. The GROUPS directive is used to form the factor.

Action with RESTRICT

Restrictions on the STRUCTURE and NOBSERVATIONS parameters are not allowed.

References

None.

Procedures Used

FTEXT and SUBSET.

Similar Procedures

None.

Example

```
CAPTION   'FEXPAND example' ; STYLE=meta
VARIATE   [VALUES=1...5] x
VARIATE   [VALUES=4, 3, 2, 1, 0] nobs
FEXPAND   STRUCTURE=x ; NOBSERVATIONS=nobs ; VARIATE=newx
PRINT     newx ; DECIMALS=0
FACTOR    [LEVELS=3 ; VALUES=3(1), 2(2)] f
TABULATE  [CLASSIFICATION=f ; PRINT=means, nobservations] \
          x ; MEANS=meantab ; NOBS=nobstab
FEXPAND   STRUCTURE=meantab ; NOBSERVATIONS=nobstab ; VARIATE=newvarx
PRINT     newvarx
TEXT      [VALUES=jan, feb, mar, apr, jan, feb, mar] month
VARIATE   [VALUES=  3,   4,   2,   5,   1,   2,   3] ntimes
FEXPAND   STRUCTURE=month ; NOBSERVATIONS=ntimes ; FACTOR=fmonth
PRINT     fmonth
TABULATE  [CLASSIFICATION=fmonth ; PRINT=counts]
FACAMEND  fmonth ; NEWLEVELS=!t(jan, feb, mar, apr)
TABULATE  [CLASSIFICATION=fmonth ; PRINT=counts]
```

FGRID

P.W. Goedhart

Forms a grid of values in one or more dimensions

contents previous next

Options

VALUES = *numerical structures*	Values from which to form a grid of values; default *
MINIMUM = *numerical structure*	Minimum value of grid in each dimension; default 0
MAXIMUM = *numerical structure*	Maximum value of grid in each dimension; default 1
NGRID = *numerical structure*	Number of grid points in each dimension; default 11

Parameters

GRID = *pointer*	To save the grid in a pointer to a set of variates
LEVELS = *pointer*	To save the distinct values of each grid variate

Description

Procedure FGRID can be used to form a grid of numerical values in one or more dimensions. The grid may be specified in either of two ways. The first method is to set the grid points in each dimension by setting the VALUES option to a list of numerical structures. The dimension then equals the number of numerical structures in the VALUES list. The second method is to specify the MINIMUM and MAXIMUM value in each dimension. NGRID then specifies the number of grid points in each dimension. In this case the dimension equals the length of MINIMUM, which must equal the length of MAXIMUM. The length of NGRID must equal 1 or the length of MINIMUM. Note that the VALUES setting takes precedence over the other options. The GRID parameter saves the grid in a pointer to a set of variates. The LEVELS parameter can be used to save the distinct values of each grid variate.

Method

The GenStat directive GENERATE is used to form the grid.

Action with RESTRICT

Restrictions on the VALUES, MINIMUM, MAXIMUM and NGRID options are not allowed.

References

None.

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION     'FGRID example' ; STYLE=meta
FGRID       [VALUES=!(1,8,2,3), !(10,49,31)] grid
PRINT       grid[]
FGRID       [MINIMUM=0 ; MAXIMUM=10 ; NGRID=11] grid
PRINT       grid[]
FGRID       [MINIMUM=!(-10,0, 100) ; MAXIMUM=!(10,1,200) ; NGRID=!(5,3,2)] \
            grid ; LEVELS=levels
PRINT       levels[], grid[]
```

FINTEGRATE

P.W. Goedhart

Calculates a definitive integral of a function of one argument

contents previous next

Options

`PRINT` = *string token*	Printed output required (`area`); default `*`
`CALCULATION` = *expressions*	Expression structures to calculate the function to be integrated
`FUNCTION` = *scalar*	Identifier of the scalar, calculated by `CALCULATION`, for which the integral must be calculated
`DATA` = *pointer*	Data to be used within the `CALCULATION` expressions or within the `%FINTEGRATE` procedure
`METHOD` = *string token*	Integration method to be used (`trapezoidal`, `simpson`, `romberg`, `legendre`); default `legendre`
`NINTERVALS` = *scalar*	Number of intervals into which the integration interval is subdivided or number of integration points for Gauss-Legendre integration; default 64
`TOLERANCE` = *scalar*	Convergence criterion for `METHOD=Romberg`; default 1.0e-4
`NOMESSAGE` = *string token*	Which message to suppress (`warning`); default `*`

Parameters

`PARAMETER` = *scalar*	Identifier of the scalar argument, used by `CALCULATION`, of the function for which the integral must be calculated
`LOWER` = *scalars*	Lower integration limit
`UPPER` = *scalars*	Upper integration limit
`AREA` = *scalars*	To save the area of the function
`SAVE` = *pointers*	To save intermediate results for `METHOD=Romberg`

Description

`FINTEGRATE` can be used to approximate the definitive integral of a function of one argument. The function for which the integral must be calculated can be defined by specifying a list of expressions with the `CALCULATION` option. The `FUNCTION` option defines the function value, say F(x), that is calculated by the expressions as a function of the `PARAMETER` parameter, say x. The `DATA` option can be employed to provide any data structures that are used in the expressions to calculate the value of the function. The `LOWER` and `UPPER` parameters must be specified to define the integration interval and the resulting area can be saved by means of the `AREA` parameter.

The `METHOD` option specifies the integration method. All methods subdivide the integration interval into a number, as specified by the `NINTERVALS` option, of subintervals of equal length. More intervals provide better approximations of the integral. The area is approximated as a weighted sum of the function values at the subinterval endpoints. For the `trapezoidal` method the first and last weights are set to 1 and the other weights are equal to 1. This is equivalent to what is used by the `AREA` function of GenStat. For the `simpson` method the first and last weights equal 1/3 and the other weights alternate between 4/3 and 2/3. The `romberg` setting uses a form of Richardson extrapolation which is iterative such that subsequent iterations provide better approximation of the integral. The iterations are abandoned when successive approximations are within the value set by `TOLERANCE`. For `METHOD=legendre` Gauss-Legendre integration is used; in this case `NINTERVALS` specifies the number of integration points.

More complicated functions can be integrated by defining a procedure `%FINTEGRATE`, and not setting the `CALCULATION` option. This has the advantage that any GenStat commands can be used to obtain the function value. The `DATA` option can be employed to provide any data structures that are needed by `%FINTEGRATE` to calculate the value of the function. Details are given in the Methods section, and the Example section provides an example.

Printed output is controlled by the `PRINT` option. The `NOMESSAGE` option can be set to suppress a warning message when the iterative procedure for `METHOD=romberg` did not converge. The `SAVE` parameter can be set for `METHOD=romberg` to save subsequent approximations of the integral in a symmetric matrix and convergence information.

Method

The trapezoidal and Simpson rules for approximating an integral are well known. Simpson's rule is generally more accurate than the trapezoidal rule. Romberg integration is even more precise and also provides an estimate of error. The implementation of Romberg's method follows closely the algorithm in http://en.wikipedia.org/wiki/Romberg's_method. The relative error of Romberg's method is calculated as the quotient of the absolute error and the current best approximation bounded by 1. The iterative scheme is abandoned when either the absolute or relative error is smaller than the tolerance. For Gauss-Legendre integration see the `GAUSSPOINTS` procedure. The skeleton of the `_FINTEGRAL` procedure, for integrating more complex functions, is given below. Note that the parameters are of type variate.

```
PROCEDURE '%FINTEGRATE'
OPTION    'DATA' ; TYPE='pointer' ; SET=no ; DECLARED=yes ; PRESENT=yes
PARAMETER 'PARAMETER', 'FUNCTION' ; TYPE='variate' ; SET=yes ; \
           DECLARED=yes,no ; PRESENT=yes,no
ENDPROCEDURE
```

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

`GAUSSPOINTS`. The subdidary procedure `%FINTEGRATE` is used.

Similar Procedures

`GAUSSPOINTS` can be used to calculate Gauss-Hermite and Gauss-Legendre integration points.

Example

```
CAPTION   'FINTEGRATE example using the CALCULATION option' ; STYLE=meta
SCALAR    mu, sigma2, lower, upper ; 0, 1, 0, 1.0
EXPRESSIO prnormal ; !e(function = PRNORMAL(argument ; mu ; sigma2))
CALCULATE aExact = NORMAL(upper ; mu ; sigma2) - NORMAL(lower ; mu ; sigma2)
POINTER   [VALUES=aTrapezoidal, aSimpson, aRomberg, aLegendre] area
FOR [INDEX=ii] method='trapezoidal','simpson', 'romberg', 'legendre'
  FINTEGRATE [CALCULATION=prnormal ; FUNCTION=function ; METHOD=#method] \
             PARAMETER=argument ; LOWER=lower ; UPPER=upper ; AREA=area[ii]
ENDFOR
PRINT      aExact, area[] ; FIELD=18 ; DECI=12
CAPTION   'FINTEGRATE example using the %FINTEGRATE procedure' ; STYLE=meta
PROCEDURE '%FINTEGRATE'
  OPTION    'DATA' ; TYPE='pointer' ; SET=yes ; DECLARED=yes ; PRESENT=yes
  PARAMETER 'PARAMETER','FUNCTION' ; TYPE='variate' ; SET=yes ; \
            DECLARED=yes,no ; PRESENT=yes,no
  DUMMY     mu, sigma2 ; DATA[1,2]
  CALCULATE FUNCTION = PRNORMAL(PARAMETER ; mu ; sigma2)
ENDPROCEDURE
POINTER   [VALUES=mu, sigma2] data
FOR [INDEX=ii] method='trapezoidal','simpson','romberg', 'legendre'
  FINTEGRATE [DATA=data ; METHOD=#method] \
             LOWER=lower ; UPPER=upper ; AREA=area[ii]
ENDFOR
PRINT      aExact, area[] ; FIELD=18 ; DECI=12
```

FISHEREXACT

P.W. Goedhart

Performs pairwise tests of independence of rows in a r x 2 table

contents previous next

Options

PRINT = *string token*	What to print (probabilities); default probabilities
METHOD = *string token*	Type of test required (twosided, lessthan, greater); default twosided
SORT = *string token*	Whether to sort the observed proportions in ascending order (no, yes); default no

Parameters

XBINOMIAL = *variates*	Observed binomial counts
NBINOMIAL = *variates*	Binomial totals
LABELS = *texts*	Text vector naming the elements of XBINOMIAL; if LABELS is unset the unit numbers of XBINOMIAL are used; default *
TWOSIDED = *symmetric matrices*	To save the tail probabilities of the test statistic corresponding to METHOD=twosided.
LESSTHAN = *symmetric matrices*	To save the tail probabilities of the test statistic corresponding to METHOD=lessthan.
GREATER = *symmetric matrices*	To save the tail probabilities of the test statistic corresponding to METHOD=greater.

Description

Fisher's exact test is an unbiased uniformly most powerful test of independence in a 2 x 2 table (Kendall and Stuart, 1979). It is particularly useful for tables with small marginal counts because the approximation of other test statistics, such as chi-squared and likelihood ratio, is poor for such tables. The test is most easily explained when one classifying factor is simply a labelling of two populations (e.g. smokers and non smokers), and the two populations are to be compared with respect to the probability of having an attribute (e.g. lung cancer). More formally, let x1 and x2 be two independent binomial distributions, x1 ~ Binomial (n1, p1) and x2 ~ Binomial (n2, p2), for which the equality of probabilities p1 and p2 is to be tested. Fisher's exact test uses the test statistic (x1 | x1+x2=r), which follows, assuming p1=p2, a hypergeometric distribution with parameters (N=n1+n2, n1, r).

The XBINOMIAL parameter specifies the observed binomial counts (x1, x2, x3, …) while NBINOMIAL specifies the binomial totals (n1, n2, n3, …). FISHEREXACT performs all pairwise tests of equality of probabilities (p1, p2, p3, …). The METHOD option, with default setting twosided, specifies which type of test is performed:

twosided	gives two-sided tests for	H_0 : pi = pj	for i < j
lessthan	gives one-sided tests for	H_0 : pi <= pj	for i < j
greater	gives one-sided tests for	H_0 : pi >= pj	for i < j

where i and j number the elements of XBINOMIAL. Twosided probabilities are calculated as the minimum of 1 and twice the smaller of the two tail probabilities. Tail probabilities can be saved in symmetric matrices TWOSIDED, LESSTHAN and GREATER. These matrices are labelled by the first 9 characters of the text vector LABELS or, if this is unset, by the unit numbers of XBINOMIAL.

The PRINT option controls the output of FISHEREXACT. By default a symmetric matrix with tail probabilities is printed with the observed percentages (100 x XBINOMIAL / NBINOMIAL) on the diagonal. The SORT option controls whether the percentages on the diagonal are sorted into ascending order. Combining SORT=yes with METHOD=lessthan is particularly useful.

Method

The standard GenStat functions `CLHYPERGEOMETRIC` and `CUHYPERGEOMETRIC` are used to calculate the hypergeometric probabilities.

Action with RESTRICT

Pairwise tests are only performed for the set of units to which `XBINOMIAL` is restricted. Restrictions on `NBINOMIAL` and `LABELS` are ignored.

References

Kendall, M. and Stuart, A. (1979). *The advanced theory of statistics, Volume 2, 4th edition*. Griffins. London.

Procedures Used

None.

Similar Procedures

Procedure `FEXACT2X2` provides an alternative way of performing Fisher's exact test.

Example

```
CAPTION     'FISHEREXACT example' ; STYLE=meta
VARIATE     Improved, Total ; VALUES=!(0,1,1,2,12),  !(100,100,10,5,20)
TEXT        [VALUES=Drug1, Drug2, Drug3, Placebo, Drug4] Labels
FISHEREXACT XBINOMIAL=Improved ; NBINOMIAL=Total ; LABELS=Labels
```

FMINIMIZE

P.W. Goedhart

Minimizes a function of one parameter argument by (modified) golden section search

contents previous next

Options

PRINT = *string tokens*	Printed output required (minimum, monitoring); default minimum, monitoring
CALCULATION = *expressions*	Expression structures to calculate the target function
FUNCTION = *scalar*	Identifier of the scalar, calculated by CALCULATION, whose value is to be minimized
DATA = *pointer*	Data to be used within the CALCULATION expressions or within the %FMINIMIZE procedure
METHOD = *string token*	Optimization method (brent, mbrent, goldensectionsearch); default brent
MAXCYCLE = *scalar*	Maximum number of iterations; default 50
TOLERANCE = *scalar*	Convergence criterion; default 1.0e-4
NOMESSAGE = *string token*	Which message to suppress (warning); default *
_PRINT = *texts*	To modify the printed monitoring of the iterative procedure); default *

Parameters

PARAMETER = *scalars*	Identifier of the scalar argument, used by CALCULATION, of the function to be minimized
LOWER = *scalars*	Lower limit of the parameter which is used for initial bracketing of the minimum
UPPER = *scalars*	Upper limit of the parameter which is used for initial bracketing of the minimum
MINIMUM = *scalars*	To save the minimum of the function
FUNCTIONVALUE = *scalars*	To save the function value at the minimum
SAVE = *pointers*	To save (intermediate) results of the iterative procedure

Description

FMINIMIZE can be used to find the minimum of a function in a given interval, similar to the MIN1DIMENSION procedure. The minimum is found by golden section search possibly combined with parabolic interpolation. The function to be minimized can be defined by specifying a list of GenStat expressions with the CALCULATION option, similarly to the way in which functions for optimization are specified for the FITNONLINEAR directive. The FUNCTION option defines the function value that is calculated by the expressions as a function of the PARAMETER parameter. The DATA option can be employed to provide any data structures that are used in the expressions to calculate the value of the function. The LOWER and UPPER parameters must be specified to define the interval in which to search for the minimum. The results of the minimization can be saved by means of the MINIMUM, FUNCTIONVALUE and SAVE parameters.

The minimization method can be specified by the METHOD option. Options setting brent uses the BRENT algorithm in Press et al (2002). Setting mbrent uses a modified version in which the bracketing interval is narrowed after each parabolic fit. Setting goldensectionsearch only uses golden section search without parabolic interpolation. The MAXCYCLE option sets a limit on the number of iterations. The TOLERANCE option controls the convergence criterion; convergence is reached when the bracketing interval [a,b] is such that (b-a) < (a+b)*TOLERANCE.

More complicated functions can be minimized by defining a procedure %FMINIMIZE, and not setting the CALCULATION option. This is more complicated to specify, but it has the advantage that any GenStat command can be used to obtain the function value. The DATA option can be employed to provide any data structures that are needed by %FMINIMIZE to calculate the value of the function. Details are given in the Methods section, and the Example section provides an example.

Printed output is controlled by the PRINT option. The NOMESSAGE option can be set to suppress a warning message when the iterative procedure did not converge. Printed monitoring information can be modified by setting the _PRINT option to two texts; see the source code of the procedure for details.

Method

FMINIMIZE performs a series of iterations in which three points are moved in one dimension to locate the minimum. The idea is that the two outer points should bracket the minimum, while the inner point locate it. Golden section search derives its name from the fact that the algorithm maintains the function values for triples of points whose distances form a golden ratio. Brent's (1973) modification to this algorithm uses parabolic interpolation whenever possible to speed up convergence. The procedure is a transcript of the C++ algorithm BRENT in Press et al (2002). The parabolic fit is omitted when golden section search is used. The skeleton of the %FMINIMIZE procedure, for minimizing more complex functions, is as follows

```
PROCEDURE '%FMINIMIZE'
OPTION    'FUNCTION', 'DATA' ; TYPE='scalar', 'pointer' ; \
          SET=yes,no ; DECLARED=no,yes ; PRESENT=no,yes
PARAMETER 'PARAMETER' ; TYPE='scalar' ; SET=yes ; DECLARED=no ; PRESENT=no
ENDPROCEDURE
```

The options and parameter of this procedure are identical to those of FMINIMIZE.

Action with RESTRICT

Not relevant.

References

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (2002). *Numerical recipes in C++. The art of scientific computing, 2nd edition*. Cambridge University Press. Cambridge.

Brent, R.P. (1973). *Algorithms for minimization without derivatives*. Prentice and Hall. New York.

Procedures Used

The subdidary procedure %FMINIMIZE is used.

Similar Procedures

MIN1DIMENSION finds the minimum of a function in one dimension. MINIMIZE and SIMPLEX find the minimum of a general function.

Example

```
CAPTION   'FMINIMIZE example using the CALCULATION option' ; STYLE=meta
EXPRESSIO e1 ; !e(Fun = BJ1(arg))
FMINIMIZE [CALCULATION=e1 ; FUNCTION=Fun] PARAMETER=arg ; LOWER=5 ; UPPER=9
SCALAR    order1, order2 ; 2,3
EXPRESSIO e2 ; !e(Fun = BJN(arg ; order1) + BJN(arg ; order2))
FMINIMIZE [CALCULATION=e2 ; FUNCTION=Fun ; DATA=!p(order1, order2)] \
          PARAMETER=arg ; LOWER=5 ; UPPER=9

CAPTION   'FMINIMIZE example using the %FMINIMIZE procedure' ; STYLE=meta
CAPTION   !t('Procedure returns the deviance of a negative binomial',  \
          'regression model for a fixed LOG(aggregation) parameter.', \
          'The example compares the results of FMINIMIZE with the', \
          'RNEGBINOMIAL procedure', '') ; STYLE=plain
```

```
PROCEDURE '%FMINIMIZE'
  OPTION    'FUNCTION', 'DATA' ; TYPE='scalar', 'pointer' ; \
            SET=yes,no ; DECLARED=no,yes ; PRESENT=no,yes
  PARAMETER 'PARAMETER' ; TYPE='scalar' ; SET=yes ; DECLARED=no; PRESENT=no
  GET       [SPECIAL=save]
  CALCULATE kk = EXP(PARAMETER)
  CALCULATE save['rsave'][1][2]$[9] = kk   "Sets the aggregation parameter"
  FIT       [PRINT=* ; NOMESSAGE=dis,lev,res,ali,mar,ver,df,inf] #DATA[1]
  RKEEP     [PMODEL=pm] FITTED=fit
  DUMMY     yy ; pm['y']
  CALCULATE FUNCTION = -2*SUM(LNGAMMA(yy+kk) - LNGAMMA(kk) + kk*LOG(kk) + \
            yy*LOG(fit+(yy==0)) - (yy+kk)*LOG(fit+kk) - LNGAMMA(yy+1))
ENDPROCEDURE
FACTOR    [NVALUES=10; LEVELS=2; LABELS=!t('Continuous','Standby')] mode
READ      [PRINT=* ; SETNVALUES=yes] mode,events,time
    1  5  94.320   2  1  15.720   1  5  62.880   1 14 125.760   2  3   5.240
    1 19  31.440   2  1   1.048   2  1   1.048   2  4   2.096   2 22  10.480 :
CALCULATE logtime = LOG(time)
MODEL     [DISTRIBUTION=negativebinomial ; LINK=log ; OFFSET=logtime] events
RNEGBIN   [PRINT=moni ; TOL=!(1.0e-8,1.0e-8) ; AGGREGATION=Aggregation] mode
RKEEP     FITTED=fit
DUMMY     yy,kk ; events,Aggregation
CALCULATE Deviance = -2*SUM(LNGAMMA(yy+kk) - LNGAMMA(kk) + kk*LOG(kk) + \
          yy*LOG(fit+(yy==0)) - (yy+kk)*LOG(fit+kk) - LNGAMMA(yy+1))
PRINT     Aggregation, Deviance ; FIELD=-20 ; DECI=12

POINTER   [NVALUES=1] data
FORMULA   data[1] ; !f(mode)
SCALAR    lower, upper ; LOG(0.001, 1000)
FMINIMIZE [PRINT=moni ; DATA=data] LOWER=lower ; UPPER=upper ; \
          MINIMUM=Aggregation ; FUNCTIONVALUE=Deviance
CALCULATE Aggregation = EXP(Aggregation)
PRINT     Aggregation, Deviance ; FIELD=-20 ; DECI=12
```

FPOINTER

L.C.P. Keizer & J.T.N.M. Thissen

Forms a pointer from a text structure

contents previous next

Options

SCOPE = *string token*	This allows 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); default global

Parameters

TEXT = *texts*	Names of the structures to be stored in POINTER
POINTER = *pointers*	To save the pointer structure

Description

Procedure FPOINTER can be used to form a pointer from a text structure. This is especially useful for procedure writers who retrieve information about structures in a text structure, e.g. by using procedure QPICKLIST. The strings in the TEXT parameter define the structures (identifiers) of the POINTER parameter. The SCOPE option is similar to that of the ASSIGN directive.

Method

Directive ASSIGN, with the SCOPE option set, is printed to a text structure and then executed.

Action with RESTRICT

If the TEXT parameter is restricted, the POINTER is formed from the restricted text.

References

None**.**

Procedures Used

FTEXT.

Similar Procedures

RENAMEPOINTER renames the structures of a pointer.

Example

```
CAPTION   'FPOINTER example' ; STYLE=meta
UNIT      [10]
FACTOR    Treat
READ      Treat, Time[1...3], Weight ; FREPRESENTATION=labels,4(*)
 A  91.7  12.4  44.3  41.0      B  91.7  11.3  35.4  36.5
 C  92.4   9.5  48.6  44.4      D  91.8  10.4  39.9  37.1
 E  93.1  11.2  38.1  36.0      A  91.2  13.4  42.5  43.2
 B  91.9  12.1  38.4  36.9      C  91.2  11.3  41.6  45.4
 D  92.2  11.8  39.7  33.7      E  92.9  11.7  40.0  41.9
 :
QPICKLIST [TITLE='Which variables do you want to analyse?'] \
          LIST=!t('Time[1]','Time[2]','Time[3]','Weight') ; SELECTED=select
FPOINTER  select ; pointer
TREATMENT Treat
ANOVA     [PRINT=aov] pointer[]
```

FPRODUCT

J.T.N.M. Thissen

Forms a factor with a label for every combination of other factors

contents previous next

Options

SPACE = *string token*	Whether to use spaces between the labels of the factors (yes, no); default yes

Parameters

FACTORS = *pointers* or *formulae*	Factors contributing to each product
PRODUCT = *factors*	Factors to be formed
FREPRESENTATION = *texts*	Defines how the labels of the PRODUCT factor are formed from the values of the FACTORS parameter (labels, levels, ordinals); default is to use labels of the FACTORS if available and levels otherwise
IDENTIFIER = *texts*	Whether to represent the identifier of the factors from the FACTORS parameter into the labels of the PRODUCT factor (yes or no); default * uses the identifier only for factors without labels
LABELS = *texts*	Text structure to save the labels of the PRODUCT factor

Description

Procedure FPRODUCT is a modified version of the FACPRODUCT procedure from the official Procedure Library. FPRODUCT allows a factor to be formed whose labels represent all the combinations of a list of other factors. Parameter PRODUCT specifies the identifier of the factor to store the product, and parameter FACTORS gives the list of factors from which it is to be formed. These factors can be input in either a pointer or a model formula.

The labels of the PRODUCT factor are defined by the settings of the parameters FREPRESENTATION and IDENTIFIER. The length of the FREPRESENTATION and IDENTIFIER text structures should equal the length of the FACTORS pointer, or should be equal to 1 in which case the specification is for each factor. Each string of the FREPRESENTATION text structure can be set to ordinals, levels or labels indicating the way in which the factor levels appear in the labels of the PRODUCT factor. Each string of the IDENTIFIER text structure can be set to yes or no indicating whether the corresponding factor name precedes the factor level in the label. Default is to use the identifier only for factors which have no labels.

By default the labels of the factors are separated by one or more spaces. Setting option SPACE=no suppresses all spaces. The labels of the PRODUCT factor can be saved by the LABELS parameter.

Method

The FCLASSIFICATION directive is used, if necessary, to form lists of factors whose product is to be calculated. The labels for the new factor are formed by printing the labels (or levels and whether or not with identifier) of the factors of the FACTORS parameter into one text structure. The GROUPS directive is then used to form the new factor.

Action with RESTRICT

If any of the factors is restricted, the labels will be formed only for the units not excluded by the restriction.

References

None.

Procedures Used

CHECKARGUMENTS is used to check that all the elements of the FACTORS pointer are factors. FTEXT is used to form text structures from the factors and SUBSET is used in case the factors are restricted.

Similar Procedures

FACPRODUCT forms a factor with a level for every combination of other factors.

Example

```
CAPTION   'FPRODUCT example' ; STYLE=meta
FACTOR    [NVALUES=18 ; LEVELS=!(4,20,34)] temp ; DECIMALS=0
FACTOR    [NVALUES=18 ; LABELS=!t(Male,Female)] sex
GENERATE  temp, sex, 3
VARIATE   [NVALUES=18] initweight, finalweight, tumourweight
READ      initweight, finalweight, tumourweight
  18.15 16.51 0.24     18.68 19.5  0.32     19.54 19.84 0.20
  19.15 19.49 0.16     18.35 19.81 0.17     20.68 19.44 0.22
  21.27 23.30 0.33     19.57 22.30 0.45     20.15 18.95 0.35
  18.87 22.00 0.25     20.66 21.08 0.20     21.56 20.34 0.20
  20.74 16.69 0.31     20.02 19.26 0.41     17.20 15.90 0.28
  20.22 19.00 0.18     18.38 17.92 0.30     20.85 19.90 0.17  :
FPRODUCT  FACTORS=!p(sex,temp) ; PRODUCT=sextemp
PRINT     temp, 2(sex,sextemp) ; FIELD=4(9),15 ; FREP=levels, (levels,labels)2
FPRODUCT  FACTORS=!p(sex,temp) ; PRODUCT=sextemp ; \
          FREPRESENTATION=!t(ord,lev) ; IDENTIFIER='yes'
PRINT     temp, 2(sex,sextemp) ; FIELD=4(9),15 ; FREP=levels, (levels,labels)2
RESTRICT  temp, sex, finalweight, initweight ; temp .NE. 20
FPRODUCT  FACTORS=!p(sex,temp) ; PRODUCT=sextemp
PRINT     temp, 2(sex,sextemp), initweight, finalweight ; \
          FIELD=4(9),15,2(11) ; FREP=levels, (levels,labels)2, *, *
COVARIATE initweight
TREATMENT sextemp
ANOVA     [FPROBABILITY=yes] finalweight
REST      temp, sex, finalweight, initweight
PRINT     temp, 2(sex,sextemp), initweight, finalweight ; \
          FIELD=4(9),15,2(11) ; FREP=levels, (levels,labels)2, *, *
FACTOR    [MODIFY=yes ; LABELS=!t(M,F)] sex
FPRODUCT  [SPACE=no] FACTORS=!p(sex,temp) ; PRODUCT=sextemp ; \
          IDENTIFIER='no'
PRINT     temp, 2(sex,sextemp), initweight, finalweight ; \
          FIELD=4(9),15,2(11) ; FREP=levels, (levels,labels)2, *, *
PEN       1 ; SYMBOL=sextemp
DGRAPH    finalweight ; initweight
```

FROOTFIND

P.W. Goedhart

Finds the root of a function of one parameter argument by (modified) bisection

contents previous next

Options

PRINT = *string tokens*	Printed output required (root, monitoring); default root
CALCULATION = *expressions*	Expression structures to calculate the target function
FUNCTION = *scalar*	Identifier of the scalar, calculated by CALCULATION, for which the root must be found
DATA = *pointer*	Data to be used within the CALCULATION expressions or within the %FROOTFIND procedure
METHOD = *string token*	Method to be used to find the root (bisection, ridders); default bisection
MAXCYCLE = *scalar*	Maximum number of iterations; default 100
TOLERANCE = *variate*	Convergence criterion for root and function value; default !(1.0e-4, 1.0e-6)
NOMESSAGE = *string token*	Which message to suppress (warning); default *
_PRINT = *texts*	To modify the printed monitoring of the iterative procedure); default *

Parameters

PARAMETER = *scalars*	Identifier of the scalar argument, used by CALCULATION, of the function for which the root must be found
VALUE = *scalars*	Defines the root to be found, i.e. the root is the parameter for which the function value equals this setting; default 0
LOWER = *scalars*	Lower limit of the parameter which is used for initial bracketing of the root
UPPER = *scalars*	Upper limit of the parameter which is used for initial bracketing of the root
ROOT = *scalars*	To save the root of the function
FUNCTIONVALUE = *scalars*	To save the function value at the root
SAVE = *pointers*	To save (intermediate) results of the iterative procedure

Description

FROOTFIND can be used to find the root of a function in a given interval. The root is found using bisection possibly combined with a variant on false position (or regula falsi). The function for which the root must be found can be defined by specifying a list of GenStat expressions with the CALCULATION option, similarly to the way in which functions for optimization are specified for the FITNONLINEAR directive. The FUNCTION option defines the function value, say F(x), that is calculated by the expressions as a function of the PARAMETER parameter, say x. The root z is defined by F(z) = y and the value of y can be specified by mans of the VALUE parameter The default value for the VALUE parameter equals zero. The DATA option can be employed to provide any data structures that are used in the expressions to calculate the value of the function. The LOWER and UPPER parameters must be specified to define the interval in which to search for the minimum. The results of the root finding can be saved by means of the ROOT, FUNCTIONVALUE and SAVE parameters.

The root finding method can be specified by the METHOD option. The default setting is bisection, while the ridders setting uses the ZRIDDR algorithm in Press et al (2002). The MAXCYCLE option sets a limit on the number of iterations. The TOLERANCE option controls the convergence criterion. The iterative procedure is either exited when the current bracketing interval [a,b] is such that (b-a)/Max(Abs(a+b);1) is less than the first element of the TOLERANCE option or when (F(z)-y)/Max(Abs(y);1) is smaller than the second element of the TOLERANCE option.

More complicated functions can be minimized by defining a procedure %FROOTFIND, and not setting the CALCULATION option. This is more complicated to specify, but it has the advantage that any GenStat commands can be used to obtain the function value. The DATA option can be employed to provide any data structures that are needed by %FROOTFIND to calculate the value of the function. Details are given in the Methods section, and the Example section provides an example.

Printed output is controlled by the PRINT option. The NOMESSAGE option can be set to suppress a warning message when the iterative procedure did not converge. Printed monitoring information can be modified by setting the _PRINT option to two texts; see the source code of the procedure for details.

Method

The bisection method performs a series of iterations which repeatedly bisects an interval and then selects a subinterval in which the required root must lie. Bisection is robust but slow. According to Press et al (2002) Ridders (1979) method is a powerful variant on the regula falsi which itself assumes that the function is assumed to be approximately linear in the region of interest. Ridders method is a transcript of the C++ algorithm ZRIDDR in Press et al (2002). Generally speaking Ridders method will require less function evaluations than bisection. The skeleton of the %FROOTFIND procedure, for minimizing more complex functions, is as follows

```
PROCEDURE '%FROOTFIND'
OPTION    'FUNCTION', 'DATA' ; TYPE='scalar', 'pointer' ; \
          SET=yes ; DECLARED=yes ; PRESENT=yes
PARAMETER 'PARAMETER' ; TYPE='scalar' ; SET=yes ; DECLARED=yes ; PRESENT=yes
ENDPROCEDURE
```

The options and parameter of this procedure are identical to those of FROOTFIND.

Action with RESTRICT

Not relevant.

References

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (2002). *Numerical recipes in C++. The art of scientific computing, 2nd edition*. Cambridge University Press. Cambridge.

Ridders, C.J.F. (1979). New algorithm for computing a single root of a real continuous function. *IEEE Transactions on circuits and systems*, **26**, pp 979-980.

Procedures Used

The subdidary procedure %FROOTFIND is used.

Similar Procedures

None.

Example

```
CAPTION   'FROOTFIND examples', 'Find the root of a cubic function.', '' ; \
          STYLE=meta, 2(plain)
SCALAR    constant ; 1
EXPRESSIO Polynoom ; !e(Function = (Par-2)**3 - constant)
SCALAR    Lower, Upper ; 0,7
FROOTFIND [PRINT=moni,root ; CALCULATION=Polynoom ; FUNCTION=Function ; \
          DATA=!p(constant) ; METHOD=bisection] PARAMETER=Par ; \
          LOWER=Lower ; UPPER=Upper

CAPTION   !t('Calculate the 95% likelihood ratio interval for a binomial', \
          'probability and compare with an exact method.'), '' ; STYLE=plain
SCALAR    ybin, nbin ; 12,50
CALCULATE pbin = ybin/nbin
CALCULATE LogLik = LOG(PRBINOMIAL(ybin ; nbin ; pbin))
EXPRESSIO ee ; !e(Deviance = -2*(LOG(PRBINOMIAL(ybin ; nbin ; prob)) - LogLik))
CALCULATE chi = EDCHI(0.95 ; 1)
```

```
FROOTFIND [PRINT=moni,root ; CALCULATION=ee ; FUNCTION=Deviance ; \
          METHOD=ridders] PARAMETER=2(prob) ; VALUE=chi ; LOWER=0.001,pbin ; \
          UPPER=pbin,0.999 ; ROOT=LRLower, LRUpper
BNTEST    [PRINT=* ; TEST=exact] ybin ; nbin ; LOWER=Lower ; UPPER=Upper
PRINT     LRLower, LRUpper, Lower, Upper

CAPTION   !t('Calculate the 95% likelihood ratio interval for the', \
          'effective dose in a quantal bioassay and compare this with', \
          'the so-called Fieller interval.'), '' ; STYLE=plain
PROCEDURE '%FROOTFIND'
  OPTION    'FUNCTION', 'DATA' ; TYPE='scalar', 'pointer' ; \
            SET=yes ; DECLARED=yes ; PRESENT=yes
  PARAMETER 'PARAMETER' ; TYPE='scalar' ; SET=yes ; DECLARED=yes ; PRESENT=yes
  DUMMY     Kill, Nbinomial, Dosis, Percentage, Dev0 ; DATA[]
  CALCULATE offset = LOGIT(Percentage) * (Kill.GE.0)
  CALCULATE newDosis = Dosis – PARAMETER
  MODEL     [DISTRIBUTION=binomial ; OFFSET=offset] Kill ; NBINOMIAL=Nbinomial
  FIT       [PRINT=* ; CONSTANT=omit] newDosis
  RKEEP     DEVIANCE=Dev1
  CALCULATE FUNCTION = Dev1 - Dev0
ENDPROCEDURE
VARIATE   Dosis ; !(1...10)
VARIATE   Kill ; !(0,1,3,2,5,7,8,6,9,10)
SCALAR    Nbinomial ; 10
SCALAR    Percentage ; 90
MODEL     [DISTRIBUTION=binomial] Kill ; NBINOMIAL=Nbinomial
FIT       Dosis
FIELLER   %DOSE=Percentage ; VALUE=ED ; LOWER=Lower ; UPPER=Upper
RKEEP     DEVIANCE=Dev0
POINTER   [VALUES=Kill, Nbinomial, Dosis, Percentage, Dev0] data
FROOTFIND [PRINT=moni,root ; DATA=data ; METHOD=ridders] PARAMETER=2(dummy) ; \
          VALUE=chi ; LOWER=0,ED ; UPPER=ED,20 ; ROOT=LRLower, LRUpper
PRINT     Percentage, ED, Lower, Upper, LRLower, LRUpper
```

FSUBFACTOR

L.C.P. Keizer & J.T.N.M. Thissen

Forms a factor to index the units within another factor

contents previous next

Options

METHOD = *string token*	How to index the levels of the factor (global, local). Setting local uses the order of the values of FACTOR; default global

Parameters

FACTOR = *factors*	Factor within whose levels the levels of SUBFACTOR are formed; must be set
SUBFACTOR = *factors*	To save the formed factor; must be set
GROUPS = *factors*	To save the factor of new groups if METHOD=local

Description

Procedure FSUBFACTOR can be used to index the units within the levels of the FACTOR parameter. The indexed units are saved in the factor specified by the SUBFACTOR parameter. The METHOD option defines how to index the levels of the factor. The default setting global takes each level of FACTOR in turn, and numbers the corresponding units of SUBFACTOR as 1 to the number of occurrences of that level. The setting local uses the order of the values of FACTOR. First a new factor, saved by the GROUPS parameter, is created which has a new level each time the value of FACTOR changes. The units are then indexed within the newly formed factor. The GROUPS parameter can only be saved for METHOD=local.

Method

FSUBFACTOR uses standard GenStat directives for data manipulation.

Action with RESTRICT

If the FACTOR parameter is restricted the SUBFACTOR and GROUPS factor are restricted in the same way. The indices of the SUBFACTOR and GROUPS parameter are determined by the levels of FACTOR not excluded by the restriction. SUBFACTOR and GROUPS values excluded by the restriction are set to missing.

References

None.

Procedures Used

None.

Similar Procedures

AFUNITS forms a factor to index the units of the final stratum of a design.

Example

```
CAPTION    'FSUBFACTOR example' ; STYLE=meta
FACTOR     [LEVELS=3 ; VALUES=6(3,1,2)] Blocks
FACTOR     [LEVELS=2 ; VALUES=3(1,2)2,3(2,1)] Plots
FPRODUCT   !P(Blocks, Plots) ; BlockPlots
FSUBFACTOR Blocks ; WithinB
FSUBFACTOR BlockPlots ; WithinBP
PRINT      Blocks, Plots, BlockPlots, WithinB, WithinBP
FACTOR     [LEVELS=3 ; VALUES=2(1...3),1,4(2),6(3),1,2,3] factor
FSUBFACTOR [METHOD=local] FACTOR=factor ; SUBFACTOR=subfact ; GROUPS=groups
FSUBFACTOR FACTOR=groups ; SUBFACTOR=subfactnew
PRINT      factor, groups, subfact, subfactnew
```

FUNIQUETEXT

L.C.P. Keizer & J.T.N.M. Thissen

Forms a text with unique strings from another text

contents previous next

Options

PRINT = *string token*	What to print (information); default information
STRING = *text*	Text structure of length 1 specifying the character(s) between the string of OLDTEXT and added number; default '_'
JUSTIFICATION = *string token*	How to position the numbers within the field (right, left); default right

Parameters

OLDTEXT = *texts*	Text structure whose strings must be made unique; must be set
NEWTEXT = *texts*	To save the text with the newly formed unique strings
UNIQUESTRINGS = *texts*	To save the text with the unique strings of OLDTEXT
CHECK = *scalars*	To save whether OLDTEXT is already unique (1) or not (0)

Description

Procedure FUNIQUETEXT can be used to form a text structure NEWTEXT with unique strings from an existing text structure OLDTEXT. If the NEWTEXT parameter is not specified the OLDTEXT structure is overwritten by the new text structure. The non-unique strings of the OLDTEXT parameter are extended with numbers separated by the character(s) of the STRING option, so the lengths of NEWTEXT and OLDTEXT are equal. Before determining the unique strings leading and trailing spaces are removed. If the same string occurs in OLDTEXT more than 9 times, the added numbers can be right or left justified by setting the JUSTIFICATION option. The unique strings of OLDTEXT can be saved by the UNIQUESTRINGS parameter. The CHECK parameter saves whether the OLDTEXT parameter is already unique (1) or not (0).

The default setting information of the PRINT option prints a message in case there are no duplicate strings in the OLDTEXT parameter.

Method

The procedure uses directive CONCATENATE.

Action with RESTRICT

If the OLDTEXT parameter is restricted, the NEWTEXT parameter is restricted in the same way. Values in units excluded by the restriction are not altered if NEWTEXT is unset. If NEWTEXT is set the excluded units in NEWTEXT are empty.

References

None**.**

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION     'FUNIQUETEXT example' ; STYLE=meta
TEXT        [VALUES=a,a,b,12(c),d,d,d,10(a)] letters
FUNIQUETEXT [JUSTIFICATION=right] OLDTEXT=letters ; NEWTEXT=newright
FUNIQUETEXT [JUSTIFICATION=left]  OLDTEXT=letters ; NEWTEXT=newleft
FUNIQUETEXT [STRING='..'] OLDTEXT=letters ; NEWTEXT=newdots
FUNIQUETEXT [STRING='' ; JUSTIFICATION=left] OLDTEXT=letters ; \
            NEWTEXT=newnothing ; UNIQUESTRINGS=unique
PRINT       letters, newleft, newright, newdots, newnothing ; FIELD=12
PRINT       unique ; FIELD=12
```

GAUSSPOINTS

P.W. Goedhart

Calculates the nodes and weights for Gauss-Hermite and Gauss-Legendre integration

contents previous next

Options

`METHOD` = *string token*	Which integration points to return (`hermite`, `legendre`); default `hermite`
`NPOINTS` = *scalar*	Number of integration points; default 32
`MULTIPLY` = *string token*	Whether to multiply the Gauss-Hermite integration points with the squared root of 2 and the weights with the reciprocal of the squared root of pi (`yes`, `no`); default `yes`

Parameters

`NODES` = *variate*	To save the integration points
`WEIGHTS` = *variate*	To save the weights

Description

Gauss-Hermite integration can be used to approximate a specific integral, with integration limits $-\infty$ and ∞, as follows:

$$\text{Integral}(\ F(x)\ \exp(-x^2)\ ;\ -\infty\ ;\ \infty) = \Sigma_j\ w_j\ F(x_j)$$

in which x_j are the so-called nodes and w_j are the weights. It follows that, for the normal density function φ with mean μ and variance σ^2:

$$\text{Integral}(\ F(x)\ \varphi(x)\ ;\ -\infty\ ;\ \infty) = \Sigma_j\ (w_j/\sqrt{\pi})\ F(\sqrt{2}\ \sigma\ x_j + \mu)$$

Gauss-Legendre integration can be used to approximate the following integral, with integration limits –1 and 1, as follows:

$$\text{Integral}(\ F(x)\ ;\ -1\ ;\ 1) = \Sigma_j\ w_j\ F(x_j)$$

Gauss-Hermite and Gauss-Legendre integration with N integration nodes are both exact for polynomial functions up to order (2N-1).

The integration points to return is specified by the `METHOD` option. The number of nodes must be specified by means of the `NPOINTS` option and the nodes and weights are saved by the `NODES` and `WEIGHTS` parameters. For Gauss-Hermite integration the `MULTIPLY` option can be used to modify the saved nodes and weights to enable their direct use for integration with the normal density.

Method

The procedure is a transcript of the C++ algorithms GAUHER and GAULEG in Press et al (2002).

Action with RESTRICT

Not relevant.

References

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (2002). *Numerical recipes in C++. The art of scientific computing, 2nd edition*. Cambridge University Press. Cambridge.

Procedures Used

None.

Similar Procedures

This procedure is used by the `FINTEGRATE` procedure.

Example

```
CAPTION   'GAUSSPOINTS examples' ; STYLE=meta
CAPTION   !t('First 21 moments of Normal Distribution by Gauss-Hermite', \
          'integration.') ; STYLE=minor
GAUSSPOIN [METHOD=hermite ; NPOINTS=11] nodes ; weight
CALCULATE moment[1...21] = SUM(weight * nodes**(1...21))
VARIATE   gh ; !(#moment)
VARIATE   exact ; !(0,1,2...20)
CALCULATE exact = MVINSERT(exact ; MODULO(exact;2).EQ.0)
CALCULATE exact = MVREPLACE(EXP(CUMULATE(LOG(exact))) ; 0)
CALCULATE maxdiff = MAX(ABS(gh - exact)/(exact + (exact.EQ.0)))
PRINT     gh, exact, maxdiff ; FIELD=2(14),-12 ; DECIMALS=2(0),2
CAPTION   !t('Moments of the Bivariate Normal Distribution by double', \
          'Gauss-Hermite integration.') ; STYLE=minor
SCALAR    rho ; 0.389
FGRID     [VALUES=nodes,nodes]  pnodes
FGRID     [VALUES=weight,weight] pweight
CALCULATE weight2 = pweight[1] * pweight[2]
CALCULATE aa,bb = (SQRT(1+rho) + (1,-1)*SQRT(1-rho))/2
CALCULATE nodes2[1,2] = (aa*pnodes[1,2] + bb*pnodes[2,1])
CALCULATE Ex1y1 = SUM(weight2 * nodes2[1]**1 * nodes2[2]**1)
CALCULATE Ex1y2 = SUM(weight2 * nodes2[1]**1 * nodes2[2]**2)
CALCULATE Ex1y3 = SUM(weight2 * nodes2[1]**1 * nodes2[2]**3)
CALCULATE Ex2y2 = SUM(weight2 * nodes2[1]**2 * nodes2[2]**2)
CALCULATE Ex1y5 = SUM(weight2 * nodes2[1]**1 * nodes2[2]**5)
CALCULATE Ex2y4 = SUM(weight2 * nodes2[1]**2 * nodes2[2]**4)
CALCULATE Ex3y3 = SUM(weight2 * nodes2[1]**3 * nodes2[2]**3)
VARIATE   gh ; !(Ex1y1,Ex1y2,Ex1y3,Ex2y2,Ex1y5,Ex2y4,Ex3y3)
VARIATE   exact ; !(0,0,0,1,0,3,0) + !(1,0,3,0,15,0,9)*rho + \
          !(0,0,0,2,0,12,0)*rho*rho + !(0,0,0,0,0,0,6)*rho*rho*rho
CALCULATE maxdiff = MAX(ABS(gh - exact)/(exact + (exact.EQ.0)))
PRINT     gh, exact, maxdiff ; FIELD=2(14),-12 ; DECIMALS=2(6),2
CAPTION   !t('Integration of polynomials on the interval [-1,1] by', \
          'Gauss-Legendre integration.') ; STYLE=minor
SCALAR    maxpower ; 10
GAUSSPOIN [METHOD=legendre ; NPOINTS=maxpower+1] nodes ; weight
VARIATE   [NVALUES=maxpower] approx, exact
CALCULATE polynome = 0*nodes
SCALAR    iiexact ; 0
FOR [NTIMES=maxpower ; INDEX=jj]
  CALCULATE polynome = polynome + nodes**(2*jj)
  CALCULATE approx$[jj] = SUM(weight * polynome)
  CALCULATE iiexact = iiexact + 2/(2*jj+1)
  CALCULATE exact$[jj] = iiexact
ENDFOR
CALCULATE diff = ABS(approx-exact)/exact
PRINT     approx, exact, diff ; FIELD=20,20,-12 ; DECI=15,15,2
```

GENBATCH

P.W. Goedhart & L.C.P. Keizer

Runs several GenStat programs simultaneously in batch

contents previous next

Options

PRINT = *string tokens*	What to print (information, message); default information
PROCESSORS = *scalars* or *variates*	Which processor cores to use for the GenStat programs. The number of settings determines how many GenStat programs will run at the same time; the default * employs processors 0 and 2
DIRECTORY = *text*	In which directory to save all the output and intermediate files; default *, i.e. the current working directory
CLEARDIRECTORY = *string token*	Whether to delete all files in DIRECTORY before running the programs or not (yes, no). Only relevant when DIRECTORY is set; default no
WAITSECONDS = *scalar*	The number of seconds to wait before a new job can be started; default 5
COLLECTRESULTS = *string token*	Whether results stored in individual batch jobs on backingstore channel 5 must be collected in a single backingstore file (yes, no). Only relevant when PROGRAM is set to a single-valued text; default no
DELETE = *string tokens*	Which files to delete after completion of all batch jobs (input, output, backingstore); default *, i.e. none
NDIGITS = *scalar*	Number of digits used for padding filenames with zeros; default 0, i.e. determined automatically
OUTEXTENSION = *text*	Which file extension to use for output files; default lis
OUTWIDTH = *scalar*	Width of output files; default 86
BACKEXTENSION = *text*	Which file extension to use for backingstore files; default bst
STARTSETTING = *scalar*	Number of simulation setting to start with. Only relevant when PROGRAM is set to a single-valued text; default 1
ENDSETTING = *scalar*	Number of simulation setting to end with. Only relevant when PROGRAM is set to a single-valued text; default *, i.e. the total number of simulation settings
SETTINGS = *variates*	Simulation settings to invoke. Setting this option overrides the setting of STARTSETTING and ENDSETTING. Only relevant when PROGRAM is set to a single-valued text; default *, i.e. all settings are invoked
CONTINUE = *string token*	Whether to continue from the last simulation which was successfully completed (yes, no). Only relevant when PROGRAM is set to a single-valued text; default no
DIRECTION = *string token*	Order in which to execute the programs (ascending, descending); default ascending

Parameters

PROGRAM = *texts*	Programs to run in batch; these must refer to files in the working directory
DATA = *variates, factors, texts* or *pointers*	Data to be passed when PROGRAM is set to a single-valued text
BACKINGSTOREFILES = *texts*	Saves the names of the individual backingstore files on channel 5; only relevant when PROGRAM is set to a single-valued text
COMPLETED = *variates*	Saves whether programs are completed successfully (1) or not (0)

Description

Important note: The Intel i7 processor has 4 physical cores and 4 virtual cores. Using all these cores might, after some time, results in Windows memory problems. It is therefore advised to only use the physical cores by setting the PROCESSORS option to 0, 2, 4, 6.

Procedure GENBATCH can be used to run several GenStat programs simultaneously in batch. This is especially useful for running multiple programs on a PC with multiple cores and/or threads since GENBATCH is able to utilizes all available resources. Consequently running time might be decreased considerably. It is advised to study and run the example program for a full understanding of the working of the procedure. Also some timing with different options, notably PROCESSORS and WAITSECONDS, is advised.

The number of settings of the PROCESSORS option determines how many GenStat programs will run at the same time. Each setting of PROCESSORS defines on which processor or processors a program will run. Note that the numbering of processors starts at 0, so a quad core processor has cores 0, 1, 2 and 3. A scalar setting defines a single processor, while a variate setting defines multiple processor. So for instance PROCESSORS=0, 2, !(4,5) implies that three GenStat programs will run at the same time. The first program on processor 0, the second program on processor 2 and the third program on processors 4 and 5. There are essentially two ways in which the procedure can be used.

The first and simplest way is when multiple GenStat programs must be run. Suppose that P1.gen and P2.gen are these multiple programs. The PROGRAM parameter can then be set to the text structure !t('P1.gen', 'P2.gen'). The procedure then creates intermediate GenStat files named P1-Batch.gen and P2-Batch.gen with the following extra program line at the end of the program:

```
CLOSE CHANNEL=6 ; FILETYPE=backingstore ; DELETE=yes
```

These programs are then started consecutively using a call to GenBatch.exe by means of the SUSPEND directive with the usual output file attached to output channel 1 and an intermediate backingstore file attached to the 6-th backingstore channel (see GenBatch documentation). When the number of running GenStat programs equals its maximum, as defined by the number of settings of the PROCESSORS option, the QTIMEDELAY procedure is used to wait for an output file, which is no longer used by GenBatch.exe. If the corresponding GenStat program was completed successfully, the corresponding intermediate backingstore file is deleted by the extra program line given above. In this way successful completion of every GenStat program is monitored. Since the procedure “knows” that a GenStat program is completed, the GenStat program next in line is invoked until all programs are completed.

The second way is when a single program must be called with different settings of one or more data structures, as in a simulation study. The PROGRAM parameter must then be set to this single program, say ‘Simulation.gen’, and the data structures must be defined by means of the DATA parameter. The procedure then creates a GenStat program ‘Simulation-Batch.gen’ with the extra line given above at the end of the program. Moreover, the following program lines are added to the start of the program:

```
RETRIEVE [CHANNEL=6] pointer
DUMMY    DATA[] ; pointer[]
```

Then ‘Simulation-Batch.gen’ is invoked, again by means of SUSPEND and GenBatch.exe, as many times as the number of settings, i.e. the length of the DATA structure(s). Each invocation creates an intermediate backingstore file attached to channel 6 with the current settings, i.e. the i-th invocation uses the i-th elements of the DATA structure(s). The added program lines at the start of the program then retrieve these settings as scalars or single-valued texts. Moreover the following line is added to the end of the program to check whether the invocation was completed successfully.

```
STORE  [CHANNEL=5 ; METHOD=overwrite ; SUBFILE=GenBatchCompleted] 1
```

This backingstore file on channel 5 can also be used to store results for each invocation. In the program itself there should then be a directive such as

```
STORE [CHANNEL=5 ; METHOD=replace] list of results to collect
```

In case COLLECTRESULTS=yes, these backingstore files which were attached to channel 5 are traversed in a loop, only for programs which were successfully completed, and all results are saved in pointers in a single backingstore file with the same name as the simulation program. This single backingstore file is saved in the same directory as the program file. Note that the individual backingstore files should not contain pointers and that only structures in the default subfile are accumulated. The DATA parameter can be set to a variate, factor or text when there is only one data structure, or to a pointer when there are more than one structures. In the latter case the pointer should only contain variates, factors or texts. Note that the levels of factors are used. Sometimes a single simulation is not completed successfully. In that case the

STARTSETTING and ENDSETTING options or the SETTINGS option can be used to limit the invocations. Also a previously stopped simulation can be continued by setting CONTINUE=yes. In that case the simulation will continue from the last simulation which was successfully completed (which is obtained by inspecting the backingstore files which were attached to channel 5). Note that the collection of results does not depend on these options but rather on the presence of the subfile GenBatchCompleted on the 5th backingstore channel. The DIRECTION option controls whether execution of the programs is into ascending or descending order; by default DIRECTION=ascending.

The PRINT option can be used to display a message on screen which monitors the progress of the multiple jobs, and to print information about the running time of each job. The progress of the multiple jobs is always written to a file which is named 'Simulation-Batch.000' for the second way of using GENBATCH (a simulation study), or 'SurName.000' for the first way where SurName is the surname of the GenStat program from which GENBATCH is invoked. The DIRECTORY option can be used to store all input, output and backingstore files in a separate directory and the CLEARDIRECTORY option can be set to yes to empty this directory before running the programs. The WAITSECONDS option specifies how many seconds to wait before it is checked whether an output file of the current running jobs is no longer opened by GenBatch.exe. The DELETE option, which should be used with great care, can be used to delete input, output and backingstore files. Note that the backingstore file attached to channel 6 is always deleted. Finally the OUTEXTENSION and BACKEXTENSION options can be used to specify the file extensions to use, and the OUTWIDTH option specifies the width of the output files.

When a single program is run with different settings, the output is saved in files with the surname of the program appended with a hyphen (i.e. -) followed by the numbers 1, 2, etc. By default these numbers are appended with zeros such that the number of digits equals the number of digits of the largest number. Option NDIGITS can be set to append the numbers with more digits than the default. The surnames of the backingstore files on channel 5, which are used when COLLECTRESULTS=yes, equal the surnames of the output files. The names of these backingstore files can be saved by using the BACKINGSTOREFILES parameter. Finally the COMPLETED parameter can be used to save whether each program is completed successfully (1) or not (0).

Method

The procedure uses the SUSPEND directive with option CONTINUE=yes to call GenBatch.exe repeatedly. Each call to GenBatch.exe attaches a backingstore file to channel 6, and in the second use also to channel 5. The DOS START command is used to call GenBatch.exe employing the AFFINITY option of START to specify the processors on which to run GenBatch.exe. To this end the PROCESSORS settings are converted to hexadecimal numbers. When the number of running GenStat programs equals its maximum the QTIMEDELAY procedure is invoked to wait for an output file which is no longer opened by GenBatch.exe. When the QTIMEDELAY procedure returns control to the GENBATCH procedure the next GenStat program in line is called.

Action with RESTRICT

The PROGRAM and DATA parameters should not be set to restricted structures.

References

None

Procedures Used

The following procedures are used: BIOMETRIS, DIRLIST, QMESSAGE, QTIMEDELAY and SFILENAME. The subsidiary procedures %QSTOPWATCHHLP, %GB1MULTIPLE, %GB2MESSAGE and %GB3PROGRESS are also used.

Similar procedures

None.

Example

```
CAPTION   !t('GENBATCH example:', \
          'A small simulation study with BetaBinomial data.') ; STYLE=meta
" GenStat program which performs a simulation for a single parameter set "
TEXT      simulation ; !t(\
          '" Values below are for testing the simulation program "', \
          'ENQUIRE   CHANNEL=6 ; FILETYPE=back ; OPEN=open', \
          'IF .NOT.open', \
          '  SCALAR    seed    ; 379243', \
          '  SCALAR    nvalues ; 50', \
          '  SCALAR    ntimes  ; 10', \
          '  SCALAR    alfa    ; 0.2', \
          '  SCALAR    beta    ; 0.2', \
          '  SCALAR    nbin    ; 3', \
          'ENDIF', \
          'CALCULATE init = URAND(seed ; 1)', \
          'VARIATE   [NVALUES=nvalues] simbeta, yy', \
          'VARIATE   [NVALUES=ntimes] constant, phi', \
          'FOR [NTIMES=ntimes ; INDEX=ii]', \
          '  CALCULATE sbeta = GRBETA(nvalues ; alfa ; beta)', \
          '  CALCULATE yy = 0', \
          '  CALCULATE #nbin(yy) = yy + (URAND(#nbin(0);nvalues).LT.sbeta)', \
          '  MODEL     [DISTRIBUTION=binomial] yy ; NBINOMIAL=nbin', \
          '  RBETABINO [PRINT=moni ; ESTI=esti ; PHI=sphi ; METHOD=brent]', \
          '  CALCULATE (constant,phi)$[ii] = esti,sphi', \
          'ENDFOR', \
          'PRINT     constant, phi', \
          'ENQUIRE   CHANNEL=5 ; FILETYPE=back ; OPEN=open', \
          'IF open', \
            'STORE     [CHANNEL=5 ; METHOD=replace] constant, phi', \
          'ENDIF')
OPEN      'Simulation.gen' ; CHANNEL=2 ; FILETYPE=output
PRINT     [CHANNEL=2 ; IPRINT=* ; SQUASH=yes] simulation ; JUSTIFICATION=left
CLOSE     CHANNEL=2 ; FILETYPE=output
" Settings for single simulations "
POINTER   [VALUES=seed, nvalues, ntimes, alfa, beta, nbin] data
READ      [SETNVALUES=yes] data[]
          873289321   50   10   0.2  0.2    3
          783274921   50   10   0.2  0.5    3
          087814312   50   10   0.2    1    3
          519852382   50   10   0.2    2    3
          149892344   50   10   0.5  0.5    3
          998132838   50   10   0.5    1    3
          628384124   50   10   0.5    2    3
          098103811   50   10     1    1    3
          132457923   50   10     1    2    3
          236386543   50   10     2    2    3
          :
GENBATCH  [DIRECTORY='Simulation' ; COLLECTRESULTS=yes] 'Simulation.gen' ; \
          DATA=data ; COMPLETED=completed
" Retrieve stored data and compare mean of estimates with true values "
OPEN      'Simulation.bst' ; CHANNEL=2 ; FILETYPE=back
RETRIEVE  [CHANNEL=2 ; SUBFILE=data] data, completed
RETRIEVE  [CHANNEL=2] constant, phi
CLOSE     CHANNEL=2 ; FILETYPE=back
PRINT     [MISSING='-'] data[], completed ; FIELD=11
PRINT     constant[]
CALCULATE TrueCon = LOGIT(100*alfa/(alfa+beta))
CALCULATE TruePhi = 1/(1+alfa+beta)
VARIATE   [NVALUES=NVALUES(completed)] MeanCon, MeanPhi
RESTRICT  completed ; completed ; SAVESET=present
CALCULATE MeanCon$[#present] = MEAN(constant[#present])
CALCULATE MeanPhi$[#present] = MEAN(phi[#present])
PRINT     [MIS='-'] alfa, beta, TrueCon, MeanCon, TruePhi, MeanPhi ; \
          DECIMALS=2(2),4(4)
```

GMULTIVARIATE

M.J.W. Jansen, J.C.M. Withagen & J.T.N.M. Thissen

Generates pseudo-random numbers from multivariate normal or Student's t distribution

contents previous next

Options

PRINT = *string token*	Whether to print a summary (summary); default * prints no output
DISTRIBUTION = *string token*	Type of distribution required (normal, student); default normal
NVALUES = *scalar*	Number of values to generate; default 1
MEANS = *variate*	The mean for the multivariate Normal or Student's t distribution; default is a variate with values all equal to 0
VCOVARIANCE = *diagonal* or *symmetric matrix*	The variance-covariance matrix for the multivariate Normal or Student's t-distribution; default is to use an identity matrix
DF = *scalar*	Number of degrees of freedom for Student's t distribution; default *
SEED = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initialises the sequence automatically if no random numbers have been generated in this job

Parameters

NUMBERS = *pointers* or *matrices*	Saves the random numbers as either a pointer to a set of variates or a matrix

Description

Procedure GMULTIVARIATE generates pseudo-random numbers from a multivariate Normal or from a multivariate Student's t distribution. The type of distribution can be set by the DISTRIBUTION option. The mean *mu* is specified by the option MEANS as a variate of length *p*; the variance-covariance matrix *Sigma* is specified by the option VCOVARIANCE as a diagonal or symmetric matrix with *p* rows and columns; and the option NVALUES specifies the number of values to be generated. Note that VCOVARIANCE must be positive semi-definite. The DF option must be used to specify the number of degrees of freedom for the Student distribution and must be at least 3.

The SEED option can be set to initialise the random-number generator, hence giving identical results if the procedure is called again with the same options. If SEED is not set, generation will continue from the previous sequence in the program, or, if this is the first generation, the generator will be initialised by CALCULATE.

The numbers can be saved using the NUMBERS parameter, in either a pointer to a set of variates, or a matrix. If the NUMBERS structure or structures are already declared, their dimensions must be compatible with the settings of the NVALUES, MEANS and VCOVARIANCE options. The dimensions are also used, if necessary, to set defaults for the options. By default, MEANS is taken to be a variate of zero values, and VCOVARIANCE is taken to be the identity matrix. If the setting of NUMBERS is not already declared, it will be defined as a pointer to a set of variates with dimensions deduced from the option settings.

Method

Pseudo-random numbers from a multivariate Normal distribution are generated by forming a matrix Y of columns of univariate Normal random numbers, using the Box-Muller method (Box & Muller 1958), followed by a linear transformation

X = A Y + *mu*,

where A is calculated by a Choleski decomposition, AA' = *Sigma*. See, for example, Johnson (1987, pages 52-55) or Tong (1990, pages 181-186). Pseudo-random numbers from the multivariate Student distribution are generated according to the definition of the multivariate Student distribution:

t(*mu*, *Sigma*, *df*) ~ *mu* + MN(0, *Sigma*) / Sqrt(Chi-squared(*df*)/*df*)

where MN(0, *Sigma*) is multivariate normal with mean 0 and variance-covariance *Sigma*; and where the scalar Chi-squared(*df*) has a chi-square distribution with *df* degrees of freedom. See, for example, Box &

Tiao (1973). Note that the variance-covariance matrix of the multivariate Student distribution equals [*df* / (*df* - 2)] *Sigma*.

Action with RESTRICT

Variates that have been restricted will receive output from GMULTIVARIATE only in those units that are not excluded by the restriction. Values in the excluded units remain unchanged. Note that the NVALUES option must equal the full size of the variates. Restrictions on the MEANS variate are ignored.

References

Box, G.E.P. and Muller, M.E. (1958). A note on generation of normal deviates. *Annals of Mathematical Statistics*, **28**, 610-611.

Johnson, M.E. (1987). *Multivariate Statistical Simulation*. John Wiley & Sons, New York.

Tong, Y.L. (1990). *The Multivariate Normal Distribution*. Springer-Verlag, New York.

Box, G.E.P. & Tiao, G.C. (1973). *Bayesian inference in statistical analysis*. John Wiley & Sons, New York.

Procedures Used

None.

Similar Procedures

GROBINOMIAL generates pseudo-random numbers from an overdispersed binomial distribution. GROPOISSON generates pseudo-random numbers from an overdispersed Poisson distribution GRMULTINORMAL generates pseudo-random numbers from a multivariate normal distribution.

Example

```
CAPTION       'GMULTIVARIATE example' ; STYLE=meta
VARIATE       [VALUES=1,2,3] mean
SYMMETRIC     [ROWS=3 ; VALUES=1, 0,4, 1,3,9] vcov
GMULTIVARIATE [NVALUES=100 ; MEANS=mean ; VCOVARIANCE=vcov ; SEED=52] norm
GMULTIVARIATE [PRINT=summary ; DISTRIBUTION=student ; NVALUES=100 ; \
              MEANS=mean ; VCOVARIANCE=vcov ; DF=10 ; SEED=52] stud
DSCATTER      norm[]
DSCATTER      stud[]
```

GROBINOMIAL

P.W. Goedhart

Generates pseudo-random numbers from an overdispersed binomial distribution

contents previous next

Options

`DISTRIBUTION` = *string token*	Distribution for which pseudo-random numbers must be generated (`betabinomial`, `logitnormal`); default `betabinomial`
`LINK` = *string token*	Link function for logit-normal distribution (`complementaryloglog`, `logit`, `probit`); default `logit`
`SEED` = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initialises the sequence automatically if no random numbers have been generated in this job

Parameters

`PROBABILITY` = *variates*	Probability of the overdispersed binomial distribution; must be set
`NBINOMIAL` = *scalars* or *variates*	Number of binomial trials; must be set
`DISPERSION` = *scalars* or *variates*	Dispersion parameter of overdispersed binomial distribution; must be set
`NUMBERS` = *variates*	Saves the random numbers from the distribution

Description

`GROBINOMIAL` can be used to generate random number from two overdispersed binomial distributions.

The beta-binomial distribution arises by assuming that the probability p of success follows a Beta(α, β) distribution and that conditionally on p the data are binomially(n, p) distributed in which n is the number of binomial trials. The mean and variance of the beta-binomial are given by $n\pi$ and $n\pi(1\text{-}\pi)(1 + \varphi(n\text{-}1))$ respectively with probability π= $\alpha/(\alpha+\beta)$ and dispersion parameter $\varphi=1/(1+\alpha+\beta)$. The back-transformation is given by $\alpha=\pi(1\text{-}\varphi)/\varphi$ and $\beta=(1\text{-}\pi)(1\text{-}\varphi)/\varphi$. The parameterisation used here is in terms of the probability π and the dispersion parameter φ.

The logit-normal distribution arises by assuming that the logit of the probability p follows a normal distribution with mean logit(π) and a variance equal to the dispersion parameter φ. The resulting logit(p) is then back-transformed to the probability scale and conditionally on p the data are again binomially distributed. Alternative link functions for the logit-normal distribution can be specified by means of the `LINK` option.

The `DISTRIBUTION` option specifies the required distribution. The parameters `PROBABILITY`, `NBINOMIAL` and `DISPERSION` specify the parameters of the distribution. The random numbers can be saved by means of the `NUMBERS` parameter. The `SEED` option can be set to initialise the random-number generator, hence giving identical results if the procedure is called again with the same options. If `SEED` is not set, generation will continue from the previous sequence in the program, or, if this is the first generation, the generator will be initialised by GenStat.

Method

The beta-binomial distribution is obtained by mixing the binomial distribution with the Beta distribution. The GenStat commands to generate beta-binomial random numbers are thus basically given by

```
CALCULATE nvalues = NVALUES(PROBABILITY)
CALCULATE alfa,beta = ((0,1) + (1,-1)*PROBABILITY)*(1-DISPERSION)/DISPERSION
CALCULATE grbeta = GRBETA(nvalues ; alfa ; beta)
CALCULATE NUMBERS = GRBINOMIAL(nvalues ; NBINOMIAL ; grbeta)
```

The GenStat commands to generate logit-normal random numbers are given by

```
CALCULATE grnormal = ILOGIT(GRNORMAL(nvalues ; LOGIT(100*PROBABILITY) ; \
          DISPERSION))/100
CALCULATE NUMBERS = GRBINOMIAL(nvalues ; NBINOMIAL ; grnormal)
```

Action with RESTRICT

The NUMBERS variate will only receive random numbers for those units to which PROBABILITY is restricted. Values in the excluded units remain unchanged. Restrictions on the NBINOMIAL and DISPERSION parameters are not allowed and restrictions on the NUMBERS variate are ignored.

References

None.

Procedures Used

None.

Similar Procedures

GROPOISSON generates pseudo-random numbers from an overdispersed Poisson distribution.

Example

```
CAPTION     'GROBINOMIAL examples for beta-binomial distribution' ; STYLE=meta
CALCULATE   init = URAND(93901 ; 1)
SCALAR      prob, nbin, disp ; 0.1, 100, 0.2
VARIATE     [VALUES=100000(#prob)] prob1
GROBINOMIAL [DISPERSION=disp] prob1 ; NBINOMIAL=nbin ; NUMBERS=sample
CALCULATE   MeanExact = nbin*prob
CALCULATE   SdExact = SQRT(nbin*prob*(1-prob) * (1 + disp*(nbin-1)))
CALCULATE   MeanSample = MEAN(sample)
CALCULATE   SdSample   = SD(sample)
PRINT       MeanExact, MeanSample, SdExact, SdSample ; DECIMALS=4
VARIATE     [VALUES=10(0.1,0.9)] prob2
GROBINOMIAL [DISPERSION=0.5] prob2 ; NBINOMIAL=100 ; NUMBERS=numbers
PRINT       prob2, numbers ; DECIMALS=1,0
```

GROPOISSON

P.W. Goedhart

Generates pseudo-random numbers from an overdispersed Poisson distribution

contents previous next

Options

`DISTRIBUTION` = *string token*	Distribution for which pseudo-random numbers must be generated (`overdispersed`, `negbinomial`, `lognormal`); default `negbinomial`
`SEED` = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initialises the sequence automatically if no random numbers have been generated in this job

Parameters

`MU` = *variates*	Mean of the overdispersed Poisson distribution; must be set
`DISPERSION` = *scalars* or *variates*	Overdispersion parameter of the overdispersed Poisson distribution; must be set
`NUMBERS` = *variates*	Saves the random numbers from the distribution

Description

A negative binomial distribution can be obtained by mixing the gamma distribution with the Poisson distribution, i.e. by assuming that the Poisson mean itself follows a gamma distribution. McCullagh and Nelder (1989) describe two ways to do so:

1. The mean of the Poisson distribution follows a gamma distribution with mean μ and index parameter $\mu\nu$. This results in what is called here the overdispersed Poisson distribution which has mean μ and variance $\mu(1+\nu)/\nu$. This distribution thus has a variance which is proportional to the mean. This distribution can be obtained by `DISTRIBUTION=overdispersed` and the `DISPERSION` option specifies the parameter $\varphi = (1+\nu)/\nu$.
2. The mean of the Poisson distribution follows a gamma distribution with mean μ and index parameter $1/\nu$. This results in the negative binomial distribution which has mean μ and variance $\mu+\nu\mu^2$. This distribution can be obtained by `DISTRIBUTION=negbinomial`; the `DISPERSION` option specifies the parameter ν.

A third way to generate overdispersed Poisson data is related to generalized linear mixed models.

3. In this case the logarithm of the mean of the Poisson distribution is generated by a normal distribution with mean λ and variance parameter σ^2. Using $\sigma^2 = \mathrm{Log}(\nu+1)$ and $\lambda = \mathrm{Log}(\mu) - \sigma^2/2$ results in a distribution with the same mean μ and variance $\mu+\nu\mu^2$ as the negative binomial distribution. Note that probabilities and other moments are not the same.. This distribution can be obtained by `DISTRIBUTION=lognormal`; the `DISPERSION` option specifies the parameter ν.

Procedure `GROPOISSON` can be used to generate pseudo-random numbers from either distribution as specified by the `DISTRIBUTION` option. The possibly different means μ are specified by the `MU` parameter. The random numbers can be saved by means of the `NUMBERS` parameter. The `SEED` option can be set to initialise the random-number generator.

Method

The GenStat commands to generate the overdispersed Poisson random numbers are basically given by

```
CALCULATE nrandom = NVALUES(MU)
CALCULATE index = 1/(DISPERSION-1)
CALCULATE gamma1 = GRGAMMA(nrandom ; MU*index ; 1/index)
CALCULATE overdispersedpoisson = GRPOISSON(nrandom ; gamma1)
CALCULATE gamma2 = MU*GRGAMMA(nrandom ; 1/DISPERSION ; DISPERSION)
CALCULATE negativebinomial = GRPOISSON(nrandom ; gamma2)
CALCULATE sigma2 = LOG(DISPERSION + 1)
CALCULATE lambda = LOG(MU) - sigma2
CALCULATE meanPoisson = GRLOGNORMAL(nrandom ; lambda ; sigma2)
CALCULATE lognormal = GRPOISSON(nrandom ; meanPoisson)
```

Action with RESTRICT

The NUMBERS variate will only receive random numbers for those units to which MU is restricted. Values in the excluded units remain unchanged. Restrictions on the NUMBERS variate are ignored, and restrictions on the DISPERSION parameter are not allowed.

References

McCullagh, P and Nelder, J.A. (1989). *Generalized linear models, second edition*. Chapman and Hall. London.

Procedures Used

None.

Similar Procedures

GROBINOMIAL generates pseudo-random numbers from an overdispersed binomial distribution.

Example

```
CAPTION    'GROPOISSON examples' ; STYLE=meta
SCALAR     index ; 2
VARIATE    [VALUES=100000(10)] mu
CALCULATE  init = URAND(842982 ; 1)
GROPOISSON [DISTRIBUTION=overdispersed] mu ; DISPERSION=index ; \
           NUMBERS=overdispersed
GROPOISSON [DISTRIBUTION=negbinomial  ] mu ; DISPERSION=index ; \
           NUMBERS=negbinomial
GROPOISSON [DISTRIBUTION=lognormal    ] mu ; DISPERSION=index ; \
           NUMBERS=lognormal
PRINT      MEAN(overdispersed, negbinomial, lognormal) ; FIELD=20
PRINT      VARIANCE(overdispersed, negbinomial, lognormal) ; FIELD=20
VARIATE    [VALUES=10(5,100)] mu2
VARIATE    [VALUES=5(10,1)2] index2
GROPOISSON mu2 ; DISPERSION=index2 ; NUMBERS=numbers2
PRINT      mu2, index2, numbers2 ; DECIMALS=0
```

GUNITCUBE

M.J.W. Jansen, J.C.M. Withagen & J.T.N.M. Thissen

Generates pseudo-random numbers from a distribution with marginal uniform distributions

contents previous next

Options

NVALUES = *scalar*	Number of values to generate; default 1 or deduced from the NUMBERS parameter
RCORRELATION = *scalar* or *symmetricmatrix*	Required rank correlation matrix of multivariate distribution; default is the identity matrix
SEED = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initializes the sequence automatically if no random numbers have been generated in this job
STRATIFICATION = *string token*	Stratification (none, latin); default none
METHOD = *string token*	Method to achieve rank correlation (simple, iman); default simple

Parameters

NUMBERS = *pointers* or *matrices*	Saves the random numbers as either a pointer to a set of variates or a matrix

Description

Procedure GUNITCUBE generates pseudo-random numbers from a multivariate distribution with marginal distributions that are uniform on the interval from 0 to 1, and with a given rank-correlation matrix RCORRELATION. The numbers can be saved using the NUMBERS parameter, in either a pointer to a set of variates, or a matrix. If the NUMBERS structures are already declared, their dimensions must be compatible with the settings of the NVALUES and RCORRELATION options. Otherwise the dimensions of the NUMBERS pointer are deduced from these options. The dimensions of NUMBERS are also used, if necessary, to set defaults for the options. If NUMBERS is not declared in advance, RCORRELATION must be set. By default RCORRELATION is taken to be the identity matrix. If the setting of NUMBERS is not already declared, it will be defined as a pointer to a set of variates with dimensions deduced from the option settings.

An ordinary random sample is obtained by the option settings STRATIFICATION=none and METHOD=simple. Option setting STRATIFICATION=latin can be used to obtain Latin-hypercube samples, with marginal sample distributions that are very nearly uniform, while option setting METHOD=iman imposes close resemblance between the sample correlation matrix and RCORRELATION.

If RCORRELATION is set, the required rank correlation will be introduced according to the specified METHOD option (thus, METHOD has no effect if RCORRELATION is unset). The combination of RCORRELATION set to an identity matrix and METHOD=simple is stochastically equivalent to RCORRELATION unset.

To avoid values very close to 0 and 1, NUMBERS smaller than 0.000005 and larger than 0.999995 are set to these respective limits.

Method

The method to construct a latin hypercube sample stems from McKay et al (1979). The method to introduce the required rank correlation stems from Iman & Conover (1982).

Action with RESTRICT

Any restrictions on variates of the NUMBERS pointer will be cancelled and all units will be used.

References

Iman, R.L. & Conover, W.J. (1982). A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics - Simulation and Computation*, **11(3)**, 311-334.

McKay, M.D. & Beckman, R.J. & Conover, W.J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**, 239-245.

Procedures Used

None.

Similar procedures

None.

Example

```
CAPTION    'GUNITCUBE example' ; STYLE=meta
SCALAR     nvariates, nvalues, seed ; VALUE=3, 100, 93746
SYMMETRIC  [ROWS=nvariates] corr
CALCULATE  corr = DIAGONAL(!(#nvariates(1)))
CALCULATE  corr$[2,3;1] = -0.8, 0.4
GUNITCUBE  [NVALUES=nvalues ; RCORRELATION=corr ; SEED=seed ; \
           STRATIFICATION=latin ; METHOD=iman] uni
PRINT      MEAN(uni[])
PRINT      VARIANCE(uni[])
CORRELATE  [PRINT=correlations] uni[]
CAPTION    'Marginal distributions are nearly uniform', ' '
GROUPS     uni[1...3] ; funi[1...3] ; LIMITS=!(0.1,0.2...0.9)
TABULATE   [CLASSIFICATION=funi[1] ; COUNT=count[1]]
TABULATE   [CLASSIFICATION=funi[2] ; COUNT=count[2]]
TABULATE   [CLASSIFICATION=funi[3] ; COUNT=count[3]]
PRINT      [SERIAL=yes] count[]
DSCATTER   uni[]
```

IRCLASS

A. Keen

Fits a generalized linear mixed model to grouped response variables, e.g. to ordinal data

contents previous next

Options

PRINT = *string tokens*	Printed output required (model, components, effects, means, stratumvariances, monitoring, vcovariance, Waldtests, deviance); default components, effects, stratumvariances
UDISTRIBUTION = *string token*	Underlying distribution (logistic, normal, evleft, evright, lognormal, student); default logistic
DF = *scalar*	Degrees of freedom for DISTRIBUTION=student; default * i.e. the number of degrees of freedom is estimated
CUTPOINTS = *scalars*	Fixed values for the cutpoints; default *
MULINK = *string token*	Link function relating the mean of the underlying distribution to the linear predictor (identity, logarithm, power); default identity
EXPONENT = *scalar*	Exponent for power link; no default, i.e. must be set
UDISPERSION = *scalar*	Value of dispersion of the underlying distribution; default 1
INTERCEPT = *string token*	How to treat constant (estimate, omit); default estimate
CADJUST = *string token*	What adjustment to make to covariates before analysis (mean, none); default mean
FIXED = *formula*	Fixed effects model for the mean of the underlying distribution; default *
RANDOM = *formula*	Random effects model for the mean of the underlying distribution; default *
ABSORB = *factor*	Absorbing factor; default *
INITIAL = *scalars*	Initial values for variance components; default *
CONSTRAINTS = *string token*	How to constrain each variance component (positive, fixrelative, fixabsolute); default positive
FDISPERSION = *formula*	Fixed effects model for the logarithm of the standard deviation of the underlying distribution
RDISPERSION = *formula*	Random effects model (with fixed variance component) for the log standard deviation of the underlying distribution
IDISPERSION = *scalars*	Initial value for the variance of the parameters of RDISPERSION; default 0.1
PSE = *string tokens*	Standard errors to be printed with tables of effects and means (differences, estimates, alldifferences, allestimate, none); default differences
VCCONVERGENCE = *scalar*	Variance component for fixed non-linear effects to be used as a tool for improving convergence; default 1000
MAXITER = *scalar*	Maximum number of REML iterations; default 50
MAXCYCLE = *scalar*	Maximum number of cycles within each REML; default 5

Parameters

YCOUNTS = *pointer*	Pointer of variates with numbers of observations in each of the classes
YCLASS = *variate*	Response variate of observed class numbers for each unit
FITTEDVALUES = *pointer*	To save fitted frequencies in the different classes if YCOUNTS is set and expected class number if YCLASS is set
UMEANS = *variate*	To save estimated means of the underlying distribution
LINEARPREDICTOR = *variate*	To save the linear predictor, i.e. means at the link scale
VCOVARIANCE = *symmetric matrix*	To save the variance-covariance matrix for the estimates of the variance components
ALLVCOVARIANCE = *symmetric*	To save the variance-covariance matrix for the full set of fixed and

matrix	random effects not associated with the absorbing factor
PREDICTIONS = *pointer*	To save estimates of random effects
SEPREDICTIONS = *pointer*	To save estimates of standard errors of the predictions

Description

A generalized linear mixed model (GLMM) is specified for the underlying variable in a threshold model. For ordinal data, when cutpoints are estimated, the residual variance is fixed at a default value, e.g. 1 for the normal distribution. In case cutpoints are known, the residual variance is estimated. In addition to the more traditional models with normal random effects, many other distributions with extra shape parameters may be fitted to the data. For instance, the residual variance may be modelled on the log scale, to allow for variance heterogeneity.

Most options of IRCLASS are similar to the options of GenStat directives MODEL, VCOMPONENTS and REML. The PRINT option is similar to the PRINT option of REML, however monitoring information is always printed. The PSE option is as in REML. Because the only relevant distribution is the multinomial one, option DISTRIBUTION of the MODEL directive is omitted. Option UDISTRIBUTION of the IRCLASS procedure defines the composite link function that links the mean of observed counts to the mean of the underlying distribution. UDISTRIBUTION therefore replaces option LINK of the MODEL directive, which is not used in IRCLASS, to avoid confusion. Option MULINK links the mean of the underlying distribution to the linear predictor scale. Included is the power link for continuous distributions defined at the positive axis, like the lognormal distribution. No link function is allowed for distributions defined on the whole real axis. Underlying distributions allowed are the logistic, normal, evleft, evright, lognormal and the student distribution. Ev in evleft and evright stands for extremevalue, evleft with heavy left tail and evright with heavy right tail. The first three distributions are related to the logit, probit and complementaryloglog link functions that can be specified with the LINK option of the MODEL directive. The standard deviations of the normal, logistic, extremevalue and the Student distribution with DF (>2) degrees of freedom equal 1, *pi*/sqrt(3), *pi*/sqrt(6) and sqrt(DF/(DF-2)) respectively. For the student distribution with DF equal to 1 or 2 no standard deviation exists. DF for the student distribution can be fixed by option DF. If this option is unset, DF is estimated. The model for the mean can be specified by means of options INTERCEPT, CADJUST, FIXED, RANDOM, INITIAL and CONSTRAINTS, as in VCOMPONENTS and REML. Option INTERCEPT replaces CONSTANT, to avoid confusion with CONSTRAINTS, which has the first four characters in common. The setting CONSTRAINTS=none is not possible as it is in VCOMPONENTS. The default here for CONSTRAINTS is positive. ABSORB is as in VCOMPONENTS.

For ordinal data the cutpoints are unknown and have to be estimated. For grouped data cutpoints can be set by option CUTPOINTS. If cutpoints are known the dispersion of the underlying distribution is estimated.

FDISPERSION, RDISPERSION and IDISPERSION can be used to specify the linear model for the residual variance (dispersion) at the log scale, FDISPERSION for fixed effects, RDISPERSION for random effects. Only a list of factors is allowed for FDISPERSION and RDISPERSION. The scaling is such, that the dispersion for the first level of the factors in FDISPERSION or RDISPERSION is the basic dispersion, which can be set by UDISPERSION. The model can only be specified at the log scale. The random part of the model is meant to represent a relaxation of the assumption of constant dispersion and can only be specified with fixed variance components, which can be set by IDISPERSION with default value 0.1. Because for some combinations of factor levels the amount of information to estimate the specific dispersion can be very low, it is advised to use simple models for FDISPERSION only, e.g. just one factor, and to use it mainly to check constancy of dispersion. For the same reason it may be advantageous to specify RDISPERSION and not FDISPERSION, but note that fixed effects are identical to random effects with a very large component of variance. If FDISPERSION or RDISPERSION has been specified, estimates of the parameters involved, with standard errors, are reported. Also after the analysis with fixed dispersion and after the first iteration for the full model specified (including heterogeneity of dispersion), results are printed.

MAXITER, MAXCYCLE and VCCONVERGENCE can be used to guide the iteration process. However, this should hardly ever be necessary, because precautions have been taken to prevent convergence problems. MAXITER and MAXCYCLE restrict the number of REML analyses and the number of cycles within each

REML analysis respectively. `VCCONVERGENCE` sets the component of variance used for estimating the changes in the non-linear parameters (the cutpoints, `DF` or the parameters in `FDISPERSION` and/or `RDISPERSION`) during the iteration process.

The response can be specified in two ways: as a pointer of variates containing the frequencies in the different classes by parameter `YCOUNTS`, or as a variate containing the class numbers by parameter `YCLASS`. Note that this differs from the settings required for specifying a GLM for ordinal data with GenStat using `MODEL` and `FIT`.

Method

The algorithm is developed from considering the relationship between the observed multinomial distribution of counts and the underlying distribution as a generalized linear mixed model with a composite link function resulting from the choice of underlying distribution. In case of a GLM iterative reweighted least squares (IRLS) can be applied to the adjusted dependent variate. Because of the equivalence of the Poisson distribution conditional on the multinomial total and the multinomial distribution, Poisson weights can be used for the frequencies. This then results in maximum likelihood estimates of the parameters. The extension from GLM to GLMM has been described in Engel and Keen (1994), Schall (1991) and as PQL in Breslow and Clayton (1993). The method replaces IRLS by iterative reweighted REML. In this case, with grouped data, the composite link function involves non-linear parameters, namely the unknown cutpoints (for ordinal data, otherwise the dispersion), the degrees of freedom of the Student distribution and parameters of the model for the dispersion. These non-linear parameters are estimated essentially by the Gauss Newton method, as described by Pregibon (1980). In this way in fact they are treated as location parameters for the observed frequencies. So all parameters are estimated simultaneously, avoiding difficulties with conditional accuracy's. This extension for ordinal data has been explained in Keen and Engel (1994). Within `IRCLASS` all non-linear parameters are considered as essentially random, with known variance component. They are updated in each step using the predictions of the differences with respect to the previous values. After convergence the value of the variance component (set by `VCCONVERGENCE`) is set to 1000, which means that effectively these non-linear parameters are considered as fixed effects. The random non-linear parameters in `RDISPERSION` are not updated, but they are just included in the model as linearized random deviations from the assumed average value. Their use is to improve the estimates of the fixed effects and their standard errors.

The algorithm involves two or three steps. In the first step the fixed effects model including only the unknown cutpoints as non-linear parameters to be estimated, is fitted using cumulative frequencies, assuming independent binomial distributions and omitting the last class. This preliminary analysis provides initial estimates of the linear predictor and the cutpoints. In the second step the GLMM in the mean of the underlying distribution is fitted with constant dispersion to the frequency data. If a model for the dispersion has been specified, it is fitted in the third step, extending the model used in the second step with the model for the heterogeneity of dispersion.

Action with RESTRICT

Restrictions on the response variates or the model terms are not allowed.

References

Breslow, N.E. and Clayton, D.G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 9-25.

Engel, B. and Keen, A. (1994). A simple approach for the analysis of generalized linear mixed models. *Statistica Neerlandica*, **48**, 1-22.

Keen, A. and Engel, B. (1997). Analysis of a mixed model for ordinal data by iterative re-weighted REML. *Statistica Neerlandica*, **51**, 129-144.

Pregibon, D. (1980) Goodness of link tests for generalized linear models. *Applied Statistics*, **29**, 15-24.

Schall, R. (1991). Estimation in generalized linear models with random effects. *Biometrika*, **78**, 719-728.

Procedures Used

None.

Similar Procedures

For ordinal data IRCLASS is an extension of a generalized linear model with the multinomial distribution. The way of extending a GLM to a GLMM is similar to procedures IRREML and GLMM, which are meant for other data types. For grouped data IRCLASS is an extension of the GenStat DISTRIBUTION directive.

Example

```
CAPTION   'IRCLASS example', !t('Data from Gilmour, Anderson and Rae', \
          '(1985), Biometrika 72, 593-599'), ' ' ; STYLE=meta, 2(plain)
UNIT     [34]
READ     yr, b1, b2, b3, k[1,2,3], tot
  1  1  0  0 52 25  0  77   1  1  0  0 49 17  1  67   1  1  0  0 50 13  1  64
  1  1  0  0 42  9  0  51   1  1  0  0 74 15  0  89   1  1  0  0 54  8  0  62
  1  1  0  0 96 12  0 108   1 -1  1  0 57 52  9 118   1 -1  1  0 55 27  5  87
  1 -1  1  0 70 36  4 110   1 -1  1  0 70 37  3 110   1 -1  1  0 82 21  1 104
  1 -1  1  0 75 19  0  94   1 -1 -1  0 17 12 10  39   1 -1 -1  0 13 23  3  39
  1 -1 -1  0 21 17  3  41  -1  0  0  1 37 41 23 101  -1  0  0  1 47 24 12  83
 -1  0  0  1 46 25  9  80  -1  0  0  1 79 32 11 122  -1  0  0  1 50 23  5  78
 -1  0  0  1 63 18  8  89  -1  0  0 -1 30 20  9  59  -1  0  0 -1 31 33  3  67
 -1  0  0 -1 28 18  4  50  -1  0  0 -1 42 27  4  73  -1  0  0 -1 35 22  2  59
 -1  0  0 -1 33 18  3  54  -1  0  0 -1 35 17  4  56  -1  0  0 -1 26 13  2  41
 -1  0  0 -1 37 15  2  54  -1  0  0 -1 36 14  1  51  -1  0  0 -1 63 20  3  86
 -1  0  0 -1 41  8  1  50  :
FACTOR    [LEVELS=34 ; VALUES=1...34] sire
GROUPS    yr ; FACTOR=factor ; LEVELS=levels
CALCULATE nlevels = NVALUES(levels)
VARIATE   [MODIFY=yes ; VALUES=1...nlevels] levels
IRCLASS   [UDISTRIBUTION= normal ; FIXED=yr + b1 + b2 + b3 ; RANDOM=sire ; \
          FDISPERSION=factor ; ABSORB=sire] k ; VCOVARIANCE=vcov ; \
          ALLVCOVARIANCE=all ; PREDICTIONS=trandom
PRINT     vcov
PRINT     trandom[]
PRINT     all
```

IRREML

A. Keen & B. Engel

Fits a generalized linear mixed model (GLMM)

contents previous next

Options

`PRINT` = *string tokens*	Printed output required (`model`, `components`, `effects`, `means`, `stratumvariances`, `monitoring`, `vcovariance`, `waldtests`, `deviance`, `fullmonitoring`); default `model`, `comp`, `stra`
`DISTRIBUTION` = *string token*	'Residual' distribution of the response variate, i.e. distribution conditional on the random effects (`normal`, `poisson`, `binomial`, `negativebinomial`, `gamma`, `inversenormal`, `lognormal`); default `norm`
`LINK` = *string token*	Link function (`canonical`, `identity`, `logarithm`, `logit`, `probit`, `complementaryloglog`, `reciprocal`, `power`, `squareroot`); default `loga` for `DIST=logn` or `nega` and `cano` for the other distributions, i.e. `iden` for `DIST=norm`, `loga` for `DIST=pois`, `logit` for `DIST=bino`, `reci` for `DIST=gamm`, `powe` for `DIST=inve`
`EXPONENT` = *scalar*	Exponent for power link; default -2
`PROBMIN` = *scalar*	Fixed lower bound for the binomial probability; default 0
`DISPERSION` = *scalar*	Value of dispersion parameter in calculation of s.e.s. etc.; default `*` for `DIST=norm`, `nega`, `gamm`, `inve` and default `1` for `DIST=pois`, `bino`
`WEIGHTS` = *variate*	Variate of fixed prior weights for weighted regression (apart from the iterative weights resulting from the specified distribution and link function); default `*`
`OFFSET` = *variate*	Offset variate to be included in the linear predictor; default `*`
`GLM` = *formula*	Model for initial GLM, to obtain starting values for the linear predictor; default `*` i.e. the fixed effects as specified by the `FIXED` option.
`STARTMEAN` = *variate*	Initial fitted values to be used in the first step of the REML iterations; default `*`
`FIXED` = *formula*	Fixed effects; default `*`
`RANDOM` = *formula*	Random effects; default *
`FACTORIAL` = *scalar*	Limit on the number of factors or covariates in each fixed term; default 3
`ABSORB` = *factor*	Defines the absorbing factor; default `*` i.e. none
`CONSTANT` = *string token*	How to treat the constant (`estimate`, `omit`); default `esti`
`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 `*`
`INITIAL` = *scalars*	Initial values for each variance component; default `*`
`CONSTRAINTS` = *string token*	How to constrain each variance component (`positive`, `fixrelative`, `fixabsolute`); default `posi`
`PSE` = *string tokens*	Standard errors to be printed with tables of effects and means (`none`, `differences`, `estimates`, `alldifferences`, `allestimates`); default `diff`
`RMETHOD` = *string token*	Whether to use all the random terms, just random terms in the final stratum or both when saving `RESIDUALS` (`final`, `all`, `both`); default `final`. Corresponding fitted values are produced, e.g. for `final` the fitted values are a combination of the estimated fixed effects and predicted random effects.
`MAXITER` = *scalar*	Maximum number of REML analyses; default 50

`MAXCYCLE` = *scalar*	Maximum number of iterations within each REML; default 10
`DESIGN` = *string token*	Whether or not the design is balanced (`balanced`, `unbalanced`); default `unbalanced`
`CONVERGENCE` = *scalar*	Constant between 0 and 1 to use as fixed value in the update between REML analyses; default 1, and adaptation to lower values after a number of iterations, if necessary
`CRITERION` = *scalar*	Convergence criterion; the mean relative change in the linear predictor between successive fits, expressed as a percentage; default 0.01 (%)
`ZWEIGHTS` = *string token*	Whether to fix weights to 1 in REML step (`fix`, `free`, `no`); default `no`
`EPS` = *scalar*	Value used to force the mean to keep clear from its limits; default 0.0001
`CHECK` = *string token*	Whether the adequacy of the variance function has to be checked (`yes`, `no`); default `no`
`METHOD` = *string token*	Indicates whether to use the standard Fisher scoring algorithm or the AI algorithm with sparse matrix methods (`Fisher`, `AI`); default `AI`

Parameters

`Y` = *variates*	Response variates
`NBINOMIAL` = *variates* or *scalars*	Total numbers for `DIST=bino`
`ITERATIVEWEIGHTS` = *variates*	To save iterative weights which are used at the lowest stratum in the final analysis
`YADJUSTED` = *variates*	To save the adjusted dependent variate for which the final analysis is carried out
`LINEARPREDICTOR` = *variates*	To save fitted values on the link scale
`FITTEDVALUES` = *pointers* or *variates*	To save fitted values on the original scale as specified by the `RMETHOD` option; pointers if `RMETHOD=both` and variates otherwise
`RESIDUALS` = *pointers* or *variates*	To save Pearson residuals on the original scale as specified by the `RMETHOD` option; pointers if `RMETHOD=both` and variates otherwise. The residuals are based on the means conditional upon the random effects, i.e. (y-*mu*)/sqrt(w), where *mu* is the conditional mean and w is the iterative weight (if specified, prior weights are included as well)

Description

Procedure `IRREML` fits a GLMM, applying iterative reweighted REML to the link adjusted dependent variate. A GLMM can be looked at in two different ways: as an extension of a linear mixed model (LMM), allowing the response variate to be non-normally distributed and specification of the LMM at another scale, or as an extension of a generalized linear model (GLM), allowing random terms to be added to the fixed terms in the linear predictor. The estimation procedure can be derived as an extension of the iterative re-weighted least squares algorithm for GLMs (Schall, 1991; Engel and Keen, 1994) replacing weighted least squares by weighted REML, or by approximation of the likelihood equations by iterative use of Laplace integration (penalized quasi-likelihood; Breslow and Clayton, 1993; Engel, 1997).

The options which will be most frequently used are `DISTRIBUTION` and `LINK` to specify the distribution of the response variate and the link function, and `FIXED` and `RANDOM` to specify the fixed and random part of the model. The response variate is specified by means of the `Y` parameter and, if `DIST=binomial`, binomial totals have to be specified by means of the `NBINOMIAL` parameter. The other parameters allow saving of results after the model has been fitted.

Most of the options of `IRREML` are similar to the options of GenStat directives `MODEL`, `VCOMPONENTS` and `REML`. Options `DISTRIBUTION`, `LINK`, `EXPONENT`, `DISPERSION`, `OFFSET` and `WEIGHTS` originate from the `MODEL` directive. Note that in the output the dispersion factor is presented in the form of a residual variance together with the variance components. However, the dispersion factor is part of the conditional variance, while the variance components involve the conditional mean. The lognormal distribution is added to `IRREML`. For the negative binomial distribution the default `LINK` is `logarithm`, not `canonical.` An extra option related to the binomial distribution is PROBMIN, with which a fixed lower bound for the binomial probability can be set. The binomial link function is modified accordingly.

Options FIXED, ABSORB, CONSTANT, CADJUST, RELATIONSHIP, RANDOM, INITIAL and CONSTRAINTS originate from the VCOMPONENTS directive. The setting CONSTRAINTS=none is not possible as it is in VCOMPONENTS. Here the default is CONSTRAINTS=positive. The constraint for the residual variance must not be set with CONSTRAINTS, but with DISPERSION.

Options PRINT, FACTORIAL, PSE and RMETHOD originate from the REML directive. The PRINT option is extended with the fullmonitoring setting, which provides more detailed information about the fitting process. Also, RMETHOD is extended to save two kinds of residuals and fitted values: including and excluding predictions of random parameters. However, residuals saved with RMETHOD=all will probably be meaningless for non Normal distributions.

Options specific to IRREML arre GLM, STARTMEAN, MAXITER, MAXCYCLE, DESIGN, CONVERGENCE, CRITERION, ZWEIGHTS, EPS and CHECK. The GLM option specifies a fixed effects model to be fitted in order to obtain starting values for the iterative REML. STARTMEAN can be used to specify fitted values which are used in the first REML step and can reduce computing time if a second analysis is required with a slightly modified model. MAXITER and MAXCYCLE restrict the number of REML analyses and the number of iterations within each REML analysis respectively. The CONVERGENCE option can be set to a smaller value if convergence is a problem, CRITERION can be used to set the convergence criterion and EPS specifies how close the linear predictor is allowed to approach its limits (see the Method section). Because suitable defaults have been chosen it will usually not be necessary to specify GLM, MAXITER, MAXCYCLE, CONVERGENCE, CRITERION and EPS. Only in situations with very little information in the data they might be useful. ZWEIGHTS can be used to define equal weights at the link scale. ZWEIGHTS overrules the weights set implicitly by DISTRIBUTION and LINK. DESIGN can be used to perform ANOVA in balanced designs with constant weights at the link scale instead of REML. CHECK performs a check on the adequacy of the variance function. The METHOD option determines which algorithm to use.

VKEEP can be used after IRREML in the usual way. Residuals and fitted values from VKEEP are obtained with RMETHOD=all. The PREDICT directive can not be used.

Method

First a GLM is fitted with the model as set by the GLM option, or the FIXED model otherwise. Then, for estimating the variance components, weighted REML is carried out iteratively. Weights are the usual iterative weights of a GLM and are functions of the estimated conditional means *mu*, in which predicted random effects are included.

Numerical problems may arise if the mean *mu* of an observation is not estimable, e.g. for the binomial distribution if the binomial fraction equals 0 or 1, which corresponds with plus or minus infinity for logit(*mu*). In order to avoid these problems, the linear predictor is restricted such that means remain within a distance EPS from their limits. A message is printed when these limits are reached.

Differences between successive fits in the iteration process are characterized by differences in the linear predictor. The mean relative change in the linear predictor, expressed as a percentage, is therefore used as convergence criterion. Convergence problems may occur if information about effects is poor, inducing large changes from one step to the next, possibly leading to a constant change between alternating estimates. This is remedied automatically, using a proportion *alfa* of the new linear predictor plus a proportion (1-*alfa*) of the previous linear predictor for the next step in case estimates tend to alternate. When the iteration process diverges, the iteration is restarted with a lower value for *alfa*. The CONVERGENCE option can be used to fix the value of *alfa*.

The extra parameter *phi* in the variance function of the negative binomial distribution ($mu + phi\ mu^2$) is estimated by extending Williams' method. This involves equating Pearson's Chisquare statistic to an approximate number of degrees of freedom. These degrees of freedom are obtained by subtracting approximate degrees of freedom for the random components from the total residual degrees of freedom after fitting the fixed effects. Approximate degrees of freedom for a random term are calculated as the ratio of the sum of squared predictions of the random effects and the corresponding estimated variance component, see also Engel et al. (1995).

The `CHECK` option produces a plot of absolute residuals to the power 2/3 against the linear predictor. Taking absolute values of residuals allows looking at a trend in means in stead of a trend in ranges. The power 2/3 has been chosen, because the distribution of the residuals is then more symmetric.

Action with RESTRICT

The response variates and variates and factors in `GLM`, `FIXED`, `RANDOM` and `Y` may be restricted. Restrictions on different structures must be in line. The analysis is restricted accordingly.

References

Breslow, N.E. and Clayton, D.G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 9-25.

Engel, B. and Keen, A. (1994). A simple approach for the analysis of generalized linear mixed models. *Statistica Neerlandica*, **48**, 1-22.

Engel, B., Buist, W. and Visscher, A. (1995). Inference for threshold models from the generalized linear mixed model perspective. *Genetics Selection Evolution*, **27**, 15-32.

Engel, B. and Keen, A. (1996). Contribution to the discussion of Lee and Nelder (1996) Hierarchical generalized linear models, *Journal of the Royal Statistical Society, series B*, **58**, 656-657.

Engel, B. (1997). *Extending Generalized Linear Models with Random Effects and Components of Dispersion*. Ph.D. thesis. Agricultural University Wageningen.

Schall, R. (1991). Estimation in generalized linear models with random effects. *Biometrika*, **78**, 719-728.

Procedures Used

None.

Similar Procedures

`GLMM` analyses a GLMM in essentially the same way and `VSEARCH` helps search through models for a GLMM. `IRCLASS` fits a generalized linear mixed model to ordinal data.

Example

```
CAPTION   'IRREML example', !t('Data from Engel (1986), Statistica', \
          'Neerlandica, 40, 21-33.'), ' ' ; STYLE=meta, 2(plain)
UNIT      [60]
READ      [SETNVALUES=yes] y
   3 7 3 6 7 1 8 3 6 7 8 7 12 9 14 7 8 8 5 9 1 3 1 8 18 5 3 17 7 11 12
   9 3 4 7 3 13 5 5 6 4 15 16 11 8 10 3 2 4 4 9 22 6 7 11 8 12 9 4 7  :
FACTOR    [LEVELS=3] Location
FACTOR    [LEVELS=2] Method, Pattern
FACTOR    [LEVELS=5] Replicat
GENERATE  Method, Pattern, Location, Replicat
IRREML    [PRINT=monitoring, effects, means, components, waldtest ; \
          DISTRIBUTION=poisson ; FIXED=Location*Method*Pattern ; \
          RANDOM=Location.Method.Replicat] Y=y ; FITTEDVALUES=Mu
IRREML    [PRINT=monitoring, effects, means, components, waldtest ; \
          DISTRIBUTION=gamma ; LINK=logarithm ; \
          FIXED=Location*Method*Pattern ; \
          RANDOM=Location.Method.Replicat] Y=y ; FITTEDVALUES=Mu
```

LRPAIR

P.W. Goedhart

Gives (scaled) likelihood ratio tests for all pairwise differences of means from a regression or GLM

contents previous next

Options

PRINT = *string tokens*	What to print (teststatistics, probabilities); default test, prob

Parameters

TREATFACTOR = *factors*	Factors for which to perform tests of all pairwise differences
TESTSTATISTICS = *symmetric matrices*	To save the (scaled) likelihood ratio test statistics (missing values on the diagonal)
PROBABILITIES = *symmetric matrices*	To save the probabilities of the test statistics (missing values on the diagonal)

Description

When analysing a (generalized) linear model with the regression directives FIT, ADD etc., effects of factors in the model may be assessed from an analysis of variance (or deviance) table. With the PREDICT directive tables of estimated means and their standard errors can be obtained. The LRPAIR procedure provides additional information on such tables by calculating (scaled) likelihood ratio test statistics and corresponding probabilities for tests of all pairwise differences of means. This is similar to procedure RPAIR which calculates Wald test statistics instead of (scaled) likelihood ratio statistics. LRPAIR is especially useful for models in which parameter estimates are plus or minus infinity. This can e.g. happen in Poisson and binomial GLMs when some factor categories have zero counts only. Note that for ordinary regression RPAIR and LRPAIR are equivalent.

A call to LRPAIR must be preceded by fitting a (generalized) linear model. The TREATFACTOR parameter must be set to a factor for which all pairwise differences must be tested. This factor must be an additive term in the fitted model, so interactions with this factor are not allowed. In case the MODEL statement for the regression has defined multiple response variates, tests are only calculated for the first response variate.

The PRINT option controls the output. By default a symmetric matrix of pairwise test statistics is printed, as well as a symmetric matrix with corresponding probabilities. The TESTSTATISTICS and PROBABILITIES parameters can be used to save the output.

Method

Suppose the equality of the parameters associated with levels 1 and 2 of a factor must be tested. The (scaled) likelihood ratio test is then obtained by employing the deviance of the model with the factor itself, and the deviance of the model with a modified factor in which the levels 1 and 2 have been combined. Since there are in general multiple pairwise comparisons this is done in a loop. When the DISPERSION parameter of the MODEL statement is set to a fixed value, the test statistic is the difference between the two deviances scaled by the fixed dispersion, and probabilities are calculated by means of the Chi-squared distribution. In case DISPERSION=*, the deviance difference is scaled by the quotient of the deviance and the residual degrees of freedom of the full model, and the F distribution is used to calculate probabilities.

The procedure redefines the maximal model to the fitted model by means of the TERMS directive. This is due to a limitation of GenStat. This implies that the following two sets of statements produce a fault, since the maximal model after the call to LRPAIR only has terms Rows + Treatment.

```
MODEL   response
TERMS   Rows + Cols + Treatment
FIT     Rows + Treatment
LRPAIR  Treatment
FIT     Rows + Cols + Treatment
```

```
MODEL   response
FIT     Rows + Treatment
LRPAIR  Treatment
FIT     Rows + Cols + Treatment
```

This can be overcome by redefinition of the maximal model after the call to LRPAIR, or by repeating the MODEL statement.

Action with RESTRICT

TREATFACTOR must not be restricted, or it must have the same restriction as the response variate.

References

None**.**

Procedures Used

SUBSET.

Similar Procedures

Procedures RPAIR and PAIRTEST both produce a symmetric matrix of two-sided t-probabilities for tests of all pairwise differences of estimates. Procedure PPAIR displays results of tests for pairwise differences in compact diagrams.

Example

```
CAPTION   'LRPAIR example', !t('One of the Treatment categories has zero', \
          'counts only. The corresponding parameter in the Poisson', \
          'regression model is minus infinity. Pairwise testing by means', \
          'of the Wald statistic, using RPAIR, then fails. The likelihood', \
          'ratio test statistic, using LRPAIR, produces correct results.'), \
          '' ; STYLE=meta,2(plain)
UNIT      [NVALUES=25]
FACTOR    [LEVELS=5] Block
FACTOR    [LABELS=!T(K, M, N, O, P) ; LEVELS=!(11,13,14,15,16)] Treatment
GENERATE  Block, Treatment
VARIATE   [VALUES=0,4,0,2,8,1,6,0,4,6,1,3,0,3,1,5,1,0,2,2,2,8,0,1,6] Count
TABULATE  [CLASS=Treatment ; PRINT=tot] Count
MODEL     [DISTRIBUTION=poisson] Count
FIT       Block + Treatment
LRPAIR    Treatment
RPAIR     [PRINT=*] !p(Treatment) ; DIFF=diff ; TVAL=tval ; TPROB=tprob
PRINT     [RLWIDTH=2 ; SERIAL=yes] diff, tval, tprob ; FIELD=8 ; DECI=2,2,3
```

MATCHTARGET

J.T.N.M. Thissen & L.C.P. Keizer

Extracts units of a set of vectors by matching a target vector

contents previous next

Options

SORT = *string token*	Whether the values of TARGETVECTOR should be sorted (yes, no); default no
DIRECTION = *string token*	Order in which to sort (ascending, descending); default ascending

Parameters

OLDVECTORS = *pointers*	Set of vectors from which units are extracted; must be set
TARGETVECTOR = *variates* or *texts*	The target vector according to which units from OLDVECTORS are extracted; must be set
NEWVECTORS = *pointers*	Set of vectors to save the extracted units of OLDVECTORS; must be set

Description

MATCHTARGET extracts units of a set of vectors by matching a target vector and saves these units in a new set of vectors. MATCHTARGET gives control over the order of the extracted units, and is thus an alternative for the SUBSET procedure. The target vector specified by the TARGETVECTOR parameter must be a text structure or a variate with unique values. The first vector of the OLDVECTORS parameter must be of the same type as the TARGETVECTOR and must also have unique values. The other structures of OLDVECTORS can be factors, variates or text structures. The NEWVECTORS parameter saves all units of the OLDVECTORS structures for which the TARGETVECTOR equals the first vector of OLDVECTORS. In case an element of the target vector is not present in the first vector of OLDVECTORS, corresponding units in the NEWVECTORS are set to missing. The number of values of the NEWVECTORS structures is thus equal to the number of values of the TARGETVECTOR. The first structure of NEWVECTORS is always a copy of TARGETVECTOR.

By default the values of the NEWVECTORS structures are in the same order as the TARGETVECTOR. The SORT option can be used, in combination with option DIRECTION, to sort the TARGETVECTOR in ascending or descending order.

The difference with the SUBSET procedure is the order in which the units are saved, which is given by the TARGETVECTOR, and the inclusion of missing units which makes the length of the NEWVECTORS structures equal to the length of the TARGETVECTOR.

Method

The EQUATE directive, with proper specifications of the options OLDFORMAT and NEWFORMAT, is used to perform the extraction.

Action with RESTRICT

If the TARGETVECTOR is restricted, only the subset of values specified by the restriction will be included in the extraction. The OLDVECTORS structures must not be restricted.

References

None**.**

Procedures Used

SUBSET.

Similar Procedures

`WEAVEVECTORS` weaves two sets of vectors into a new set according to the first vector of both sets. `SUBSET` forms vectors containing subsets of the values in other vectors. `JOIN` joins or merges two sets of vectors together, based on the values of sets of classifying keys.

Example

```
CAPTION     'MATCHTARGET example' ; STYLE=meta
SCALAR      nyear, nvariety; 7, 21
TEXT        tvariety
READ        [PRINT= d, e] nr, tvariety, stand, year[1...nyear]
   1 Ritmo          0   10.9  10.5  11.3  10.8  10.5  10.7  10.8
   2 Hereward       1   12.0  12.0  12.5  12.1  11.6  12.1  12.2
   3 Vivant         0   11.3  10.3  11.6  10.8  10.4  10.7  10.8
   4 Bercy          1   11.8  11.6  11.9  11.5  11.1     *     *
   5 Versailles     0   10.8  11.1  11.3  10.6  10.6     *     *
   6 Arnaut         1   12.0  12.2  12.0  12.0  12.0  11.7  11.4
   7 Tambor         1   12.0  11.9  12.1  11.9  12.2     *     *
   8 Tower          0   11.6  11.6  11.8  11.6  11.4     *     *
   9 Urban          0      *     *  12.7  12.2  12.5     *     *
  10 Residence      1      *     *  11.8  11.5  11.2  11.3     *
  11 Harrier        0      *     *  11.4  10.8  10.1     *     *
  12 Riant          0      *     *     *     *     *  11.4     *
  13 Semper         1      *     *     *     *  11.3  11.8  12.2
  14 'PBIS 95/91'   0      *     *     *     *     *  11.3     *
  15 'Ceb 9607'     0      *     *     *     *     *  11.4     *
  16 'DI 304'       0      *     *     *     *     *  11.7     *
  17 'NIC 92-3533a' 0      *     *     *     *     *  11.5  11.6
  18 'DI 403'       0      *     *     *     *     *     *  11.9
  19 'DI 404'       0      *     *     *     *     *     *  12.3
  20 'VDH 1099-95'  0      *     *     *     *     *     *  11.6
  21 'NIC 94-3667B' 0      *     *     *     *     *     *  10.9
 :
VARIATE     [VALUES=6, 3, 24, 4] vtarget
MATCHTARGET [SORT=no] OLDVECTORS=!p(nr, tvariety, year[1...nyear]) ; \
            TARGETVECTOR=vtarget ; NEWVECTORS=!p(n, vtvariety, vy[1...nyear])
PRINT       n, vtvariety, vy[] ; FIELD=4, 13, #nyear(6) ; \
            DECIMALS=0, *, #nyear(1) ; JUSTIFICATION=r ,l, #nyear(r)
TEXT        [VALUES=Riant, Residence, Semper, A, Ritmo] ttarget
MATCHTARGET [SORT=y] OLDVECTORS=!p(tvariety, year[1...nyear]) ; \
            TARGETVECTOR=ttarget ; NEWVECTORS=!p(ttvariety, ty[1...nyear])
PRINT       ttvariety, ty[] ; FIELD=17, #nyear(6) ; \
            DECIMALS=*, #nyear(1) ; JUSTIFICATION=left, #nyear(right)
```

MOSTSIMILAR

J.T.N.M. Thissen

Displays the most similar units for each candidate unit given a set of variates

contents previous next

Options

PRINT = *string token*	What to print (description); default description
METHOD = *string token*	Which distance metric to use (cityblock, euclidean); default cityblock
FIELDWIDTH = *scalar*	Field width in which to print the results; default 10
DECIMALS = *scalar*	Number of decimal places for printing the results; default *

Parameters

DATA = *pointers*	Pointer to variates; must be set
UNITS = *texts*	Text structure to identify the units; must be set
YARDSTICK = *variates*	Yardsticks of the variates in DATA; must be set
GROUPS = *factors*	Factor with 2 levels: level 1 for candidate units, level 2 for reference units; by default all units are candidate units
SINGULARUNITS = *texts*	To save the singular units

Description

Procedure MOSTSIMILAR can be used to display the most similar units for each candidate unit given a set of variates as specified by the DATA pointer. The most similar units with METHOD=cityblock are the units with all absolute differences less than or equal to the corresponding value of the YARDSTICK parameter. With METHOD=euclidean the most similar units are the units within an ellipsoid around the candidate unit; the YARDSTICK parameter then defines the lengths of the axes of the ellipsoid. The length of the YARDSTICK variate must thus be equal to the number of DATA variates. The units must be identified by a text structure as specified by the UNITS parameter. The GROUPS parameter can be used to subdivide the units into candidate and reference units. The GROUPS factor must have the levels 1 and 2; level 1 for the candidate units and level 2 for the reference units. If the GROUPS factor is not set all units are considered candidate units. The set of candidate units that don't have most similar units can be saved by the SINGULARUNITS parameter.

The PRINT option can be used to omit the description, and the FIELDWIDTH and DECIMALS options both operate in a straightforward way. If the setting of FIELDWIDTH is smaller than the longest structure name, the length of the longest structure name plus 1 is taken as field width. If not set the default value is 10. If the DECIMALS option is not set the number of decimals is determined by application of the DECIMALS procedure to the YARDSTICK variate.

Method

MOSTSIMILAR uses simple calculations.

Action with RESTRICT

Restrictions are not allowed.

References

None**.**

Procedures Used

DECIMALS, SREPLACE, RENAMEPOINTER and VEQUATE.

Similar Procedures

BKEY constructs an identification tree.

Example

```
CAPTION   'MOSTSIMILAR example 1', !t('Data taken from example 6.19.1', \
          'of the HCLUSTER directive'), ' ' ; STYLE=meta, 2(plain)
TEXT      Cars
POINTER   [VALUES=CC, NCyl, Tank, Wt, Length, Width, Ht, WBase, TSpeed, \
          StSt, Carb, Drive] Vars
READ      Cars, Vars[]
      Estate  1490  4  50  966 414 161 133 245 177 10.9  1  2
     Arna1_5  1409  4  50  845 399 162 139 242 174 10.2  1  2
     Alfa2_5  2492  6  49 1160 433 163 140 251 210  8.2  1  1
   Mondialqc  3185  8  87 1430 458 179 126 265 249  7.4  2  1
  Testarossa  4942 12 120 1506 449 198 113 255 291  5.8  2  1
       Croma  1995  4  70 1180 450 176 143 266 209  7.8  2  2
       Panda   965  4  35  761 338 149 146 216 134 16.8  1  2
     Regatta  1585  4  55  970 426 165 141 244 180 10.0  1  2
    Regattad  1714  4  55  980 426 165 141 245 150 18.9  3  2
         Uno   999  4  42  720 364 155 143 236 145 16.2  1  2
         X19  1498  4  48  912 397 157 118 220 171 11.0  1  1
     Contach  5167 12 120 1446 414 200 107 245 286  4.9  1  1
       Delta  1585  4  45 1000 389 162 138 247 195  8.2  1  2
       Thema  1995  4  70 1150 459 175 143 266 224  7.6  2  2
         Y10  1049  4  47  790 339 151 143 216 179 11.8  1  2
      Spider  1995  4  45 1050 414 162 125 228 190  9.0  2  1  :
VARIATE     [NVALUES=12] yardstick
CALCULATE   yardstick$[1...12] = SQRT(VAR(Vars[1...12]))
MOSTSIMILAR [DECIMALS=1 ; METHOD=euclidean] DATA=Vars ; UNIT=Cars ; \
            YARDSTICK=yardstick
CAPTION   'MOSTSIMILAR example 2', !t('Data taken from the BKEY example,', \
          'i.e. common clinical yeasts'), ' ' ; STYLE=meta, 2(plain)
TEXT      Yeasts
FACTOR    [LABELS=!t('-','+')] C11 ; EXTRA='Maltose growth'
 &        C18 ; EXTRA='Lactose growth'
 &        C19 ; EXTRA='Raffinose growth'
 &        C36 ; EXTRA='D-Glucuronate growth'
 &        N1  ; EXTRA='Nitrate growth'
 &        V5  ; EXTRA='Growth w/o Thiamin'
 &        O2  ; EXTRA='0.1% Cycloheximide growth'
 &        E5  ; EXTRA='Splitting cells'
POINTER   [VALUES=C11,C18,C19,C36,N1,V5,O2,E5] Factors
POINTER   [VALUES=vC11,vC18,vC19,vC36,vN1,vV5,vO2,vE5] Variates
READ      [PRINT=data,errors] Yeasts, Factors[] ; FREPRESENTATION=labels
           'Candida albicans'  '+'  '-'  '-'  '-'  '-'  '+'  '+'  '-'
           'Candida glabrata'  '-'  '-'  '-'  '-'  '-'  '-'  '-'  '-'
       'Candida parapsilosis'  '+'  '-'  '-'  '-'  '-'  '+'  '-'  '-'
         'Candida tropicalis'  '+'  '-'  '-'  '-'  '-'  '+'  '+'  '-'
       'Cryptococcus albidus'  '+'   *    *   '+'  '+'  '-'  '-'  '-'
     'Cryptococcus laurentii'  '+'  '+'  '+'  '+'  '-'   *    *   '-'
  'Filobasidiella neoformans'  '+'  '-'   *   '+'  '-'  '-'  '-'  '-'
    'Issatchenkia orientalis'  '-'  '-'  '-'  '-'  '-'  '+'  '-'  '-'
    'Kluyveromyces marxianus'  '-'   *   '+'  '-'  '-'  '+'  '+'  '-'
      'Pichia guilliermondii'  '+'  '-'  '+'  '-'  '-'  '+'  '+'  '-'
       'Rhodotorula glutinis'  '+'  '-'   *   '-'  '+'   *    *   '-'
   'Rhodotorula mucilaginosa'   *   '-'  '+'  '-'   *   '-'   *   '-'
      'Trichosporon beigelii'   *   '+'   *   '+'  '-'  '-'   *   '+'  :
PRINT       [MISSING='V'] Yeasts,Factors[] ; FIELDWIDTH=27,8(4) ; DECIMALS=0
FACTOR      [MODIFY=yes ; LEVELS=!(0,1); LABELS=!t(negative,positive)] \
            Factors[]
CALCULATE   Variates[] = Factors[]
VARIATE     [VALUES=8(0)] Yardstick
MOSTSIMILAR [DECIMALS=0] DATA=Variates ; UNITS=Yeasts ; YARDSTICK=Yardstick
```

OCATTRIBUTES

J.T.N.M. Thissen

Calculates operating characteristic curves for single and multiple acceptance sampling plans for attributes

contents previous next

Options

`PRINT` = *string token*	What to print (`probabilities`); default `probabilities`
`PLOT` = *string token*	What to plot (`occurve`); default `occurve`
`TITLE` = *text*	General title for plot; default `*`
`DISTRIBUTION` = *string token*	Type of distribution (`binomial`, `hypergeometric`, `poisson`); default `binomial`
`MAXPERCENTAGE` = *scalar*	Maximum percentage of defectives for which acceptance probabilities are calculated; default 30
`STEPLENGTH` = *scalar*	Steplength between the percentages defectives for which acceptance probabilities are calculated; default 1
`POPULATIONSIZE` = *scalar*	Population size for `DISTRIBUTION=hypergeometric`; default 1000

Parameters

`SAMPLESIZE` = *scalars* or *variates*	Size of the sample(s); must be set
`ACCEPTANCENUMBER` = *scalars* or *variates*	Acceptance number(s); must be set
`REJECTIONNUMBER` = *scalars* or *variates*	Rejection number(s); must be set
`PROBABILITYACCEPTANCE` = *variates*	Saves the probabilities of acceptance
`PERCENTAGEDEFECTIVE` = *variates*	Saves the percentages defectives
`AVERAGESAMPLENUMBER` = *variates*	Saves the average sample numbers

Description

`OCATTRIBUTES` calculates operating characteristic (OC) curves for single and multiple acceptance sampling plans for attributes. The sampling plan is specified by the parameters `SAMPLESIZE`, `ACCEPTANCENUMBER` and `REJECTIONNUMBER`. The length of these parameters determines the number of stages of the plan. For a single sampling plan these parameters can also be set to scalars in which case `REJECTIONNUMBER` can be left unset. A negative `ACCEPTANCENUMBER` in stages before the last stage in a multiple sampling plan defines that the decision at the corresponding stage can only be to proceed or to reject. For the last stage in a multiple sampling plan the rejection number must be equal to the corresponding acceptance number + 1.

The type of distribution can be specified by the `DISTRIBUTION` option. The hypergeometric distribution is not available for multiple sampling plans. If `DISTRIBUTION=hypergeometric` the `POPULATIONSIZE` option must be set. The `MAXPERCENTAGE` option specifies the maximum percentage of defectives for which to calculate the probabilities of acceptance. For the binomial and poisson distribution, the step length between 0 and `MAXPERCENTAGE` can be supplied by the `STEPLENGTH` option.

By default `OCATTRIBUTES` plots an OC curve, but you can set `PLOT=*` to suppress this. The `TITLE` option allows you to supply a title for the graph. Also, unless you set option `PRINT=*`, `OCATTRIBUTES` prints the calculated acceptance probabilities and the average number of sample sizes.

The probabilities of acceptance can be saved by the `PROBABILITYACCEPTANCE` parameter, the percentages defectives by the `PERCENTAGEDEFECTIVE` parameter and the average sample numbers by the `AVERAGESAMPLENUMBER` parameter.

Method

The probability functions of GenStat are used to calculate the acceptance probabilities. For not too small sample sizes the Poisson distribution is a good approximation of the Binomial distribution with mean parameter (Samplesize * PercentageDefective). If the ratio between sample size and population size is small, the Binomial distribution is a good approximation of the Hypergeometric distribution.

Action with RESTRICT

Restrictions are not allowed.

References

Duncan, A.J. (1974). *Quality Control and Industrial Statistics, 4th Ed*. Irwin, Inc.
Montgomery, D.C. (2005). *Introduction to Statistical Quality Control, 5th Ed*. Wiley, New York.

Procedures Used

None**.**

Similar Procedures

`OCPLAN` finds a single acceptance sampling plan for attributes given two points on an OC curve.

Example

```
CAPTION      'OCATTRIBUTES example', \
             !t('Comparison of different sampling plans.', \
             'Data from Montgomery, page 654.', ''); STYLE=meta,plain
OCATTRIBUTES [PLOT=* ; DISTRIBUTION=binomial ; MAXPERCENTAGE=8; \
             STEPLENGTH=0.5] \
             SAMPLESIZE=50,100,200 ; ACCEPTANCENUMBER=1,2,4  ; \
             PROBABILITYACCEPTANCE= a[1...3]; PERCENTAGEDEFECTIVE=p[1...3]
PEN          1...3; METHOD=mono; SYMBOLS=0
XAXIS        1; TITLE= 'Lot percentage defective'
YAXIS        1; TITLE= 'Probability of acceptance'
DGRAPH       [TITLE= 'Figure 14-4 Montgomery']  Y=a[]; X=p[]; \
             DESCRIPTION='n=50, c=1', 'n=100, c=2', 'n=200, c=4'
OCATTRIBUTES [PLOT=* ; DISTRIBUTION=binomial ; MAXPERCENTAGE=8] \
             SAMPLESIZE=3(89) ; ACCEPTANCENUMBER=2,1,0  ; \
             PROBABILITYACCEPTANCE= a[1...3]; PERCENTAGEDEFECTIVE=p[1...3]
DGRAPH       [TITLE= 'Figure 14-5 Montgomery'] Y=a[]; X=p[]; \
             DESCRIPTION= 'n=89, c=2', 'n=89, c=1', 'n=89, c=0'
CAPTION      !t('Comparison of 7 stage sampling plan from Duncan Chapter 9,',\
             'and single sampling plan (found with OCPLAN).', ''); STYLE=plain
OCATTRIBUTES [DISTRIBUTION=binomial ; MAXPERCENTAGE=20] \
             SAMPLESIZE=     63,!(20,20,20,20,20,20,20) ; \
             ACCEPTANCENUMBER=5,!(0,1,3,5,8,9,10) ; \
             REJECTIONNUMBER= 6,!(4,5,6,8,10,11,11) ; \
             PROBABILITYACCEPTANCE= a[1...2]; \
             PERCENTAGEDEFECTIVE=p[1...2]
PEN          1...2; METHOD=mono; SYMBOLS=0
XAXIS        1; TITLE= 'Lot percentage defective'
YAXIS        1; TITLE= 'Probability of acceptance'
DGRAPH       Y=a[1,2]; X=p[1,2]; \
             DESCRIPTION='single sampling plan','7-stage sampling plan'
```

OCPLAN

J.T.N.M. Thissen

Finds a single acceptance sampling plan for attributes given two points on an OC curve

contents previous next

Options

PRINT = *string token*	What to print (plan); default plan

Parameters

PERCENTAGEDEFECTIVE = *variates*	Variate with two percentages defectives; default !(1,10)
PROBABILITYACCEPTANCE = *variates*	Variate with two probabilities of acceptance corresponding to the PERCENTAGEDEFECTIVE variate; default !(0.95,0.05)
SAMPLESIZE = *scalars*	Saves the sample size
ACCEPTANCENUMBER = *variates*	Saves the acceptance number

Description

OCPLAN finds a single acceptance sampling plan for attributes given two points of an operating characteristic (OC) curve. The two points are specified by the PERCENTAGEDEFECTIVE and PROBABILITYACCEPTANCE variates. The length of both variates is 2 with default settings PERCENTAGEDEFECTIVE=!(1,10) and PROBABILITYACCEPTANCE =!(0.95,0.05). The first percentage defective should be smaller than the second, whereas the first acceptance probability should at least 0.50 be greater than the second.

The sample size of the found single sampling plan can be saved by the SAMPLESIZE parameter and the acceptance number by the ACCEPTANCENUMBER parameter. The PRINT option prints the plan unless you set option PRINT=*.

Method

The calculation is done using percentage points of a Chi-squared distribution. This means that the calculation, apart from a rounding error to an integer of the SAMPLESIZE, is exact if the probability distribution of the number of defectives in a sample is Poisson.

Action with RESTRICT

Restrictions are ignored.

References

Cameron, J.M. (1974). Tables for constructing and for computing the operating characteristics of single-sampling plans. *Quality Progress*, **7**, 17-19.

Procedures Used

None**.**

Similar Procedures

OCATTRIBUTES calculates operating characteristic curves for single and multiple acceptance sampling plans for attributes.

Example

```
CAPTION   'OCPLAN example' ; STYLE=meta
OCPLAN    PERCENTAGEDEFECTIVE=!(5,16) ; PROBABILITYACCEPTANCE= !(0.90,0.05)
```

PER2MUTE

P.W. Goedhart

Forms all possible permutations of a set of values

contents previous next

Options

MAXPERMUTATIONS = *scalar*	The maximum number of permutations that should be stored; default 10000
SEED = *scalar*	Seed to generate the random numbers; default 0 continues an existing sequence or initialises the sequence automatically if no random numbers have been generated in this job
NOMESSAGE = *string token*	Which messages to suppress (warning); default *

Parameters

STRUCTURE = *identifiers*	Numerical structure (variate, table, matrix, symmetricmatrix, diagonalmatrix) whose values must be permuted
PERMUTATIONS = *pointers*	Pointer to a set of variates storing the permutations; the length of each variate equals the number of values in STRUCTURE,
NPERMUTATIONS = *scalars*	To save the number of permutations

Description

PER2MUTE forms all the permutations of the values in the STRUCTURE parameter. The input variate may contain multiple values. For example, the permutations of the numbers (1, 1, 2, 2) are as follows (1, 1, 2, 2), (1, 2, 1, 2), (1, 2, 2, 1), (2, 1, 1, 2), (2, 1, 2, 1) and (2, 2, 1, 1). The permutations are saved, as a set of variates each of length equal to the number of values in STRUCTURE, in a pointer supplied by the PERMUTATIONS parameter. The number of permutations can be saved by setting the NPERMUTATIONS parameter. The number of decimals of the STRUCTURE parameter is copied to the permutation variates.

If the number of permutations is larger than the setting of the MAXPERMUTATIONS option, but smaller than or equal to 40.000.000, a random subset of size MAXPERMUTATIONS is saved. If the number of permutations is larger than 40.000.000 the external C# takes too long. In that case the RANDOMIZE directive is used to return random permutations which are not guaranteed to be unique, i.e. duplicate permutations are possible. The SEED option can be set to initialise the random-number generator, if applicable, hence giving identical results if the procedure is called again with the same options. If SEED is not set, generation will continue from the previous sequence in the program, or, if this is the first generation, the generator will be initialised by CALCULATE. The NOMESSAGE option can be used to suppress warning messages.

Method

When the number of permutation is smaller than or equal to 40.000.000 the values are passed to an external C# .NET Framework 4 executable by using the SUSPEND directive. The C# program employs a translation of Applied Statistics Algorithm AS 179 (Berry, 1982) to calculate the permutations. The algorithm takes account of multiple values. When the number of permutations is larger than 40.000.000 the values are repeatedly randomized using the RANDOMIZE directive.

Action with RESTRICT

If the STRUCTURE parameter is restricted, only the values in the restriction set are used to form the permutations. The length of the permutation variates will equal the length of the restricted STRUCTURE.

References

Berry, J.B. (1982). Algorithm AS 179: Enumeration of All Permutations of Multi-Sets with Fixed Repetition Numbers. *Applied Statistics*, **31**, 169-173.

Procedures Used

The BIOMETRIS procedure is used to retrieve the filename of the external C# executable.

Similar Procedures

PERMUTE forms all possible permutations of the integers 1...n.

Example

```
CAPTION   'PER2MUTE example' ; STYLE=meta
VARIATE   [VALUES=1, 1, 2, 2, 3] v1 ; DECIMALS=0
VARIATE   [VALUES=1.1, 2.2, 3.3, 4.4] v2 ; DECIMALS=1
PER2MUTE  v1,v2 ; p1,p2
PRINT     p1[] ; FIELD=8
PRINT     p2[] ; FIELD=8
```

PPAIR

P.W. Goedhart, H. van der Voet & D.C. van der Werf

Displays results of t-tests for pairwise differences in compact diagrams

contents previous next

Options

PRINT = *string tokens*	What to print (items, groups); default groups
PROBABILITY = *scalar* or *symmetric matrix*	Level of significance for t-tests of all pairwise comparisons, default 0.05
SORT = *string token*	Whether the diagrams are sorted according to the values of (the diagonal of) DIFFERENCES (yes, no); default no
DIRECTION = *string token*	Order in which to sort when SORT=yes (ascending, descending); default ascending

Parameters

TPROBABILITIES = *symmetric matrices*	Probabilities of t-tests of pairwise comparisons; this parameter must be set
DIFFERENCES = *symmetric matrices*, *variates* or *tables*	Defines the ordering of TPROBABILITIES for the groups diagram; must be set for a groups diagram
LABELS = *texts*	Text vector labelling the output; if unset the row labels of TPROBABILITIES and the diagonal of DIFFERENCES are used
DIAGRAMS = *pointers*	Pointer to save text structures containing the diagrams

Description

Procedures RPAIR and PAIRTEST produce a symmetric matrix of two-sided t-probabilities for tests of all pairwise differences. Procedure PPAIR displays this matrix at a specified level of significance in two compact diagrams. This is especially useful when the number of estimates is large.

Input to PPAIR is a symmetric matrix TPROBABILITIES containing probabilities of the set of pairwise comparisons. The level of significance can be set by the PROBABILITY option. A common level is specified by a scalar, while a symmetric matrix specifies a level for each comparison separately (which may be useful for some multiple comparison methods). Output is labelled by the row labels of TPROBABILITIES. If the DIFFERENCES parameter is set to a symmetric matrix, the diagonal of this matrix is printed alongside these labels. This is especially useful if DIFFERENCES is saved by RPAIR or PAIRTEST because it then contains the estimates on the diagonal. DIFFERENCES can also be set to a variate or table. Alternatively the output can be labelled by specifying parameter LABELS.

The PRINT option controls which diagram is printed. PRINT=items produces a diagram which should be read line by line. Each item (represented by a letter) is followed by those items (again represented by letters) not significantly different from that item. When there are more than 52 items, letters are repeated. PRINT=groups produces a diagram in which items followed by a common letter are not significantly different. Such items are said to form a homogeneous group. This is similar to common underlining of items with non-significantly different estimates. In constructing this diagram the philosophy of multistage testing is followed, see the Methods section. The SORT option controls whether the printed and saved diagrams are sorted according to the values of (the diagonal of) DIFFERENCES. The DIRECTION option defines the sorting order. Diagrams can be saved by means of the DIAGRAMS parameter.

Method

The construction of the diagram for PRINT=groups is as follows. First the matrix of TPROBABILITIES is sorted according to the values of (the diagonal of) DIFFERENCES. The difference between the first and last item of the complete set of *n* items is then checked for significance. Next the first and last item of all subsets of *n*-1 consecutive items are checked, followed by all subsets of *n*-2 items, and so on. If non-significance is found between the first and last item of a subset, all items of the subset are said to form a homogeneous group and they receive the same letter. Clearly this only makes sense when the

TPROBABILITIES are sorted according to the estimates. The diagram only consists of homogeneous groups which are not a part of a larger group.

It is obvious that items in a homogeneous group can be significantly different. This is not displayed in the diagram, although a message is printed if this occurs. If there are no significant differences within homogenous groups, both diagrams essentially contain the same information; PRINT=groups then gives a more concise representation.

Action with RESTRICT

Restrictions on DIFFERENCES and LABELS are ignored.

References

None.

Procedures Used

DECIMALS dermines the number of decimals places of PROBABILITY if this is not defined at declaration.

Similar Procedures

Procedures RPAIR, LRPAIR and PAIRTEST produce a symmetric matrix of two-sided t-probabilities for tests of all pairwise differences of estimates.

Example

```
CAPTION   'PPAIR example 1', !t('Data taken from Cochran, W.G. and Cox, ', \
          'G.M. (1957). Experimental Designs, 2nd ed. Wiley. New York.', \
          'page 406. There are no significant differences within', \
          'homogenous groups'), ' ' ; STYLE=meta, 2(plain)
FACTOR    [LEVELS=2 ; VALUES=25(1,2)] Rep
FACTOR    [LEVELS=5 ; VALUES=5(1...5)2] Block
FACTOR    [LEVELS=25 ; VALUES=(1...25), (1,6...21), (2,7...22), (3,8...23), \
          (4,9...24), (5,10...25)] Variety
VARIATE   [VALUES= 6,7,5,8,6, 16,12,12,13,8, 17,7,7,9,14, 18,16,13,13,14, \
          14,15,11,14,14, 24,13,24,11,8, 21,11,14,11,23, 16,4,12,12,12, \
          17,10,30,9,23, 15,15,22,16,19] Yield
MODEL     Yield
FIT       [PRINT=accumulated] Rep/Block + Variety
RPAIR     [PRINT=*] !P(Variety) ; TPROBABILITIES=YieldPr ; DIFFERENCES=YieldDif
PRINT     [RLWIDTH=3] YieldPr ; FIELDWIDTH=7 ; DECIMALS=3
PPAIR     [PRINT=items,groups ; SORT=yes] YieldPr ; DIFFERENCES=YieldDif
CAPTION   'PPAIR example 2', !t('Comparison of unequally replicated', \
          'treatments with significant differences within homogenous', \
          'groups.'), ' ' ; STYLE=meta, 2(plain)
FACTOR    [LABELS=!t(aap, noot, mies, wim, zus, jet, vuur, gijs) ; \
          values=8(1), 4(2), 8(3), 7(4), 1(5), 2(6), 9(7), 9(8)] Label
VARIATE   [VALUES=5.40, 6.09, 3.53, 5.77, 4.04, 4.18, 5.24, 6.56, 4.89, \
          6.78, 6.00, 5.93, 7.90, 7.17, 5.58, 7.41, 7.79, 6.89, 6.14, \
          5.50, 3.66, 4.05, 6.24, 5.70, 5.13, 5.65, 4.58, 6.30, 7.59, \
          8.41, 5.26, 6.91, 5.86, 7.17, 6.87, 5.91, 5.22, 5.34, 7.25, \
          6.58, 8.19, 7.55, 9.58, 7.38, 7.72, 8.00, 7.92, 8.17] Response
MODEL     Response
FIT       Label
RPAIR     [PRINT=*] !P(Label) ; TPROBABILITIES=LabelPr ; DIFFERENCES=LabelDif
PRINT     LabelPr ; FIELDWIDTH=7 ; DECIMALS=3
PPAIR     [PRINT=items,groups] LabelPr ; DIFFERENCES=LabelDif
PPAIR     [PRINT=items,groups ; SORT=yes] LabelPr ; DIFFERENCES=LabelDif
```

QDIRECTORY

P.W. Goedhart

Returns a directory selected by means of a directory browse dialog box on screen

contents previous next

Options

`TITLE` = *text*	Single-valued text structure specifying the title of the directory browse dialog box; must be set
`ROOT` = *text*	Single-valued text structure which specifies the directory under which the user can browse for directories. The user will not be able to browse above this level. By default the entire file system (all drives, directories, and network shares) can be browsed. Note that a setting of C: is ignored, while C:/ or C:\ is not. Default `*`
`STARTDIRECTORY` = *text*	Single-valued text structure specifying the directory which will be selected by default when the dialog box is initially displayed. By default the top of the tree, as set by the `ROOT` option, will be selected. Default `*`
`EDITFIELDBOX` = *string token*	Whether to display an edit field in the dialog box, in which the user can type the name of a directory (`yes`, `no`). This name will be relative to the currently selected directory name in the browse list. This option is not available for all Windows versions. Default `no`
`CONFIRMPROMPT` = *string token*	Whether to display a confirmation message box in case the user types a name, of a directory which does not exist, in the edit field (`yes`, `no`). Default `no`
`NEWINTERFACE` = *string token*	Which interface to use (`yes`, `no`). The new interface provides the user with a larger dialog box that can be resized. Moreover the dialog box has additional capabilities including: drag and drop within the dialog box, reordering, shortcut menus, new folders, delete, and other shortcut menu commands. The new interface is not available for all Windows implementations. Default `no`
`NEWFOLDERBUTTON` = *string token*	Whether to display a "New Folder" button in the browse dialog box (`yes`, `no`). This option is not available for all Windows versions. Default `no`

Parameters

`DIRECTORY` = *variates*	Saves the selected directory; must be set

Description

Procedure `QDIRECTORY` can be used to return a directory selected by means of a directory browse dialog box on screen. This procedure uses an external WinBatch program. The title of the dialog box must be specified by the `TITLE` parameter. The `ROOT` and `STARTDIRECTORY` options determine the initial state of the dialog box. The user will not be able to browse above the `ROOT` level. Additionally a "New Folder" button and an "Edit Field" can be displayed by specifying the `NEWFOLDERBUTTON` and the `EDITFIELDBOX` options. Setting the `CONFIRMPROMPT` option displays a warning message when the user types a directory which does not exist in the "Edit Field". A new interface with additional capabilities can be used by specifying the `NEWINTERFACE` option. Note that the some of the options are not available for all Windows implementations. The selected directory is saved by the `DIRECTORY` parameter.

Method

The `SUSPEND [CONTINUE=no]` directive is used to invoke an external WinBatch executable.

Action with RESTRICT

Not relevant.

References

None**.**

Procedures Used

The `BIOMETRIS` procedure is used to retrieve the filename of the external WinBatch executable.

Similar Procedures

`DIRLIST` provides details about (wildcarded) files in a specified directory. `QFILENAME` returns a single filename selected by means of a file open dialog box on screen. `QMESSAGE` displays a message on screen. `QPICKLIST` can be used to pick one or more items from a list presented on screen. `QSTOPWATCH` can be used to display timing information. `QTEXT` can be used to obtain string(s) of a text structure from screen. `QYESNO` can be used to choose between alternatives Yes, No and Cancel on screen.

Example

```
CAPTION    'QDIRECTORY example' ; STYLE=meta
QDIRECTORY [TITLE='Example 1 of Biometris procedure QDIRECTORY'] \
           directory
PRINT      [IPRINT=*] directory ; SKIP=2
QDIRECTORY [TITLE='Example 2 of Biometris procedure QDIRECTORY' ; \
           ROOT='C:/Windows' ; STARTDIRECTORY='C:/Windows/System32'] directory
PRINT      [IPRINT=*] directory ; SKIP=2
```

QFILENAME

P.W. Goedhart

Returns a single filename selected by means of a file open dialog box on screen

contents previous next

Options

TITLE = *text*	Single-valued text structure specifying the title of the file open box; must be set
DIRECTORY = *text*	Single-valued text which specifies the default directory for the filename; default `*`, i.e. the current working directory
DEFAULTFILE = *text*	Single-valued text which specifies the default filename or file mask; default `'*.*'`, i.e. all files in the selected directory
FILETYPES = *texts*	File type selection definitions as used in the "Files of type" section of the file open box. Each text must be single-valued and must contain a single description and a single file mask separated by a "`|`" symbol; default `'All Files (*.*)|*.*'`
EXISTDIRECTORY = *scalar*	Saves whether the default DIRECTORY exists (1) or not (0)

Parameters

FILENAME = *texts*	Saves the name of the selected file including the full path
SAVEDIRECTORY = *variates*	Saves the directory of the selected file
SURNAME = *texts*	Saves the surname of the selected file, i.e. the name excluding the path, the period and the extension
EXTENSION = *texts*	Saves the extension of the selected file, excluding the leading period
SELECTED = *scalars*	Scalar to save whether a file is selected (1) or not (0)

Description

Procedure QFILENAME can be used to return a single filename selected by means of a file open dialog box on screen. This procedure uses an external WinBatch program. The title of the dialog box must be specified by the TITLE parameter. The default filename, which may be wildcarded, and the default directory can be set by means of the DEFAULTFILE and DIRECTORY options respectively. File type selection definitions, as highlighted in the "Files of type" section of the file open box, can be specified by means of the FILETYPES option. This option can be set to several single-valued texts, each of which must contain a single description and a single file mask separated by a "|" symbol. An illustration of the FILETYPES option is given in the example program. In case DEFAULTFILE is not defined as the first setting of the FILETYPES option, DEFAULTFILE is added as an extra file type.

The full name, including path, of the selected filename can be saved by means of the FILENAME parameter. Parts of the filename can be saved by means of parameters SAVEDIRECTORY, SURNAME, EXTENSION respectively. Note that only a single file can be selected. However when FILENAME is set to a list, for each element of the list a file open box is displayed. The SELECTED parameter saves whether a file is selected (1) or not (0). If no file is selected the FILENAME, SURNAME and EXTENSION parameters are set to the empty string `''`. The EXISTDIRECTORY option saves whether the default directory, as specified by the DIRECTORY option, exists (1) or not (0).

Method

The SUSPEND [CONTINUE=no] directive is used to invoke an external WinBatch executable.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

DIRLIST is used to save the full path of the current directory. The BIOMETRIS procedure is used to retrieve the filename of the External WinBatch executable.

Similar Procedures

DIRLIST provides details about (wildcarded) files in a specified directory. SFILENAME forms sub-strings of names of files opened by GenStat. QDIRECTORY returns a directory selected by means of a directory browse dialog box on screen. QMESSAGE displays a message on screen. QPICKLIST can be used to pick one or more items from a list presented on screen. QSTOPWATCH can be used to display timing information. QTEXT can be used to obtain string(s) of a text structure from screen. QYESNO can be used to choose between alternatives Yes, No and Cancel on screen.

Example

```
CAPTION   'QFILENAME example' ; STYLE=meta
TEXT      filetype[1] ; 'GenStat (*.g*)|*.g*'
TEXT      filetype[2] ; 'Data (*.dat)|*.dat'
TEXT      filetype[3] ; 'All Files (*.*)|*.*'
QFILENAME [TITLE='Example of Biometris procedure QFILENAME' ; \
          FILETYPE=filetype[] ; DEFAULT='*.g*'] filename ; directory ; \
          surname ; extension ; selected
IF selected
  PRINT   [ORIENTATION=across] filename, directory, surname, extension ; \
          JUSTIFICATION=left ; SKIP=3
ENDIF
```

QMESSAGE

P.W. Goedhart & L.C.P. Keizer

Displays a message on screen

contents previous next

Options

TITLE = *text*	Single-valued text structure specifying the title of the message screen; must be set
MESSAGE = *text*	Text structure specifying the message; default *
SECONDS = *scalar*	Number of seconds to display the message. For negative values of SECONDS, the message will be displayed until the user responds by selecting the OK button of the message; default -1
CONTINUE = *string token*	Whether to continue execution of GenStat without waiting for the disappearance of the displayed message (yes, no); default no
DELETEPREVIOUS = *string token*	Whether to delete the previous message with the same title (yes, no); default yes

Parameters

None.

Description

Procedure QMESSAGE can be used to display a message box on screen. This procedure uses an external WinBatch program. The title of the message box must be specified by the TITLE parameter, while the MESSAGE parameter specifies the message. The special strings @hrt and @tab in the MESSAGE parameter are translated into hard returns and tabs respectively. Note that separate lines in the MESSAGE parameter are displayed without hard returns unless the special string @hrt is used. However a line with spaces only is displayed as an empty line in the message box.

The SECONDS option can be used to specify the number of seconds the message will be displayed on screen, with a maximum of 3600 seconds. For negative values of SECONDS, the message will be displayed until the user responds by selecting the OK button of the message.

The CONTINUE option can be used to force GenStat to wait for the disappearance of the message. The DELETEPREVIOUS option enables automatic deletion of any windows with the same setting of the TITLE option.

Method

The SUSPEND [CONTINUE=no] directive is used to invoke an external WinBatch executable.

Action with RESTRICT

The MESSAGE parameter can be restricted.

References

None**.**

Procedures Used

The BIOMETRIS procedure is used to retrieve the filename of the External WinBatch executable.

Similar Procedures

QDIRECTORY returns a directory selected by means of a directory browse dialog box on screen. QFILENAME returns a filename selected by means of a file open box on screen. QPICKLIST can be used to pick one or more items from a list presented on screen. QSTOPWATCH can be used to display timing information. QTEXT can be used to obtain string(s) of a text structure from screen. QYESNO can be used to choose between alternatives Yes, No and Cancel on screen.

Example

```
CAPTION   'QMESSAGE example' ; STYLE=meta
QMESSAGE  [TITLE='Example 1 of Biometris procedure QMESSAGE' ; \
          MESSAGE=!t('You can use formatting characters in the message:', \
          '@hrt@hrt1.@tabFirst item@hrt2.@tabSecond item')]
QMESSAGE  [TITLE='Example 2 of Biometris procedure QMESSAGE' ; \
          MESSAGE='End of QMESSAGE example' ; SECONDS=10 ; CONTINUE=yes]
```

QPICKLIST

P.W. Goedhart & L.C.P. Keizer

Can be used to pick one or more items from a list presented on screen

contents previous next

Options

`TITLE` = *text*	Single-valued text structure specifying the title of the screen box; must be set
`SELECT` = *string token*	Whether it is possible to pick single or multiple items from the list (`single`, `multiple`); default `multiple`
`SORT` = *string token*	Whether to display an alphabetic list or not (`yes`, `no`); default `no`

Parameters

`LIST` = *texts*	Text structure with the items which can be picked; must be set
`SELECTED` = *texts*	Saves selected items from the `LIST` parameter; must be set
`VSELECTED` = *variates*	Saves a variate of the same length as the `LIST` parameter with elements 1 (selected) or 0 (not selected)

Description

Procedure `QPICKLIST` can be used to present a list of items on screen from which one or more items can be selected. This procedure employs an external WinBatch program. The items must be specified by the `LIST` parameter, and the selected strings are returned by means of the `SELECTED` parameter. Additionally the `VSELECTED` parameter can be used to save a variate of the same length as the `LIST` parameter with elements 1 and 0, indicating whether an item is chosen (1) or not (0). The strings of the `LIST` parameter must be unique, i.e. duplication of strings is not allowed.

The title of the screen box can be specified by means of the `TITLE` option. Option `SELECT` can be used to limit the selection to one item (`single`) or to allow multiple selection (`multiple`). The `SORT` option specifies whether the list should be displayed alphabetically or not.

After selecting items, you can proceed by using the Enter key or by pressing the OK button. Alternatively you can use the "Select None" button, in which case no items are selected.

Method

The `SUSPEND [CONTINUE=no]` directive is used to invoke an external WinBatch executable.

Action with RESTRICT

The `LIST` parameter can be restricted and only the strings included in the restriction set are displayed. Units of `VSELECTED` excluded by the restriction are set to missing.

References

None**.**

Procedures Used

The `BIOMETRIS` procedure is used to retrieve the filename of the External WinBatch executable.

Similar Procedures

`QDIRECTORY` returns a directory selected by means of a directory browse dialog box on screen. `QFILENAME` returns a filename selected by means of a file open box on screen. `QMESSAGE` displays a message on screen. `QSTOPWATCH` can be used to display timing information. `QTEXT` can be used to obtain string(s) of a text structure from screen. `QYESNO` can be used to choose between alternatives Yes, No and Cancel on screen.

Example

```
CAPTION   'QPICKLIST example' ; STYLE=meta
TEXT      [VALUES=Blue, Green, Red, Yellow, Purple] list
QPICKLIST [TITLE='Example of Biometris procedure QPICKLIST' ; SORT=yes ; \
          SELECT=multiple] list ; SELECTED=select ; VSELECTED=vselect
PRINT     list, vselect ; FIELD=12
PRINT     select ; FIELD=12
```

QREGISTRY

P.W. Goedhart

Can be used to retrieve registry keys from the registry

contents previous next

Options

`REGISTRY` = *string token*	Whether to retrieve information from the 32-bit or the 64-bit registry (`bit32`, `bit64`); default `bit32`
`HANDLE` = *string token*	From which part of the registry information must be retrieved (`machine`, `classes`, `users`, `current`); default `machine`

Parameters

`MAINKEY` = *texts*	Specifies the main key from which information must be retrieved
`KEY` = *texts*	Specifies the key from which information must be retrieved
`VALUES` = *texts*	Saves the value for each key

Description

Procedure `QREGISTRY` can be used to retrieve registry keys. Options `REGISTRY` and `HANDLE` specify the main part of the registry from which information must be retrieved. The settings of the `HANDLE` option refer to the following parts of the registry

`classes`	HKEY_CLASSES_ROOT
`current`	HKEY_CURRENT_USER
`machine`	HKEY_LOCAL_MACHINE
`users`	HKEY_USERS

The `MAINKEY` and `KEY` parameters further specify the subsections and keys that should be retrieved. The `VALUES` parameter saves the values of those keys. In case the main key or the key itself is not found, the values '*MAINKEY not found' or '*KEY not found' are returned.

Method

The `SUSPEND` directive is used to invoke an external C# .NET Framework 4 executable.

Action with RESTRICT

The `MAINKEY` and `KEY` parameters should not be restricted.

References

None.

Procedures Used

The `BIOMETRIS` procedure is used to retrieve the filename of the external C# executable. `SFILENAME` is also used.

Similar Procedures

None.

Example

```
CAPTION   'QREGISTRY example retrieves GenStat executable' ; STYLE=meta
QREGISTRY [HANDLE=classes] MAINKEY='.gdb' ; KEY='' ; VALUES=values
CONCATENA [NEWTEXT=genstat] values, '\\shell\\open\\command'
QREGISTRY [HANDLE=classes] genstat ; KEY='' ; VALUES=genexe
TXREPLACE genexe ; OLDSUB=' ""%1""' ; NEWSUB=''
PRINT     values, genexe ; SKIP=2,4
```

QSTOPWATCH

P.W. Goedhart

Can be used to display timing information

contents previous next

Options

`PRINT` = *string tokens*	Where to print timing information (`output`, `screen`); default `output`, `screen`
`NUMBER` = *scalar*	Specifies which of 10 stopwatches to use, must be set to an integer in the interval [0,9]; default 0
`MODE` = *string token*	What to do with the stopwatch (`start`, `continue`, `stop`); default `continue`, or `start` when the stopwatch is first used
`NTIMES` = *scalar*	To display timing information specific to a `FOR`-loop: number of times the loop is executed; default `*`
`INDEX` = *scalar*	To display timing information specific to a `FOR`-loop: number of the current time that the loop is being executed; default `*`
`TITLE` = *text*	Single-valued text structure specifying the title of the message screen; default `*`
`MESSAGE` = *text*	Text structure specifying the extra message; default `*`
`SAVE` = *pointer*	Pointer to save the various elements which are displayed

Parameters

None.

Description

Procedure `QSTOPWATCH` can be used to display timing information. The `PRINT` option determines whether timing information is displayed in the output file or in a message box on screen. The latter uses an external WinBatch program. By specifying the `NUMBER` option, a total number of 10 different stopwatches is available. The `MODE` option determines whether a stopwatch is started, continued or stopped. Default is to `continue` the current stopwatch, or to `start` the stopwatch when it is called the first time.

Setting `MODE=start` will print or display the starting time. `MODE=continue` will print or display the time elapsed since the stopwatch was started. `MODE=stop` will print the time elapsed since the stopwatch was started, and will remove any timing messages from screen. When `NTIMES` and `INDEX` are specified in a `FOR`-loop, the index of the current loop is displayed, and also the time elapsed and an estimate of the remaining time necessary to finish the loop. The latter assumes that the stopwatch was started just before the `FOR`-loop was started, and that each loop takes the same amount of time. The `MESSAGE` option is always displayed in the message box, but only printed to output when `NTIMES` or `INDEX` are not set. The `TITLE` option can be used to specify the title of the message screen; this only takes effect when a stopwatch is started. The `SAVE` option can be used to save the starting time of the stopwatch, the elapsed time, and for use in a `FOR`-loop, the index of the current loop and an estimate for the time remaining to finish the loop.

Method

The timing functions of GenStat are used. The `WORKSPACE` directive is employed to store timing information between successive calls to `QSTOPWATCH`. The message box is displayed by the `QMESSAGE` procedure which activates an external WinBatch executable by means of the `SUSPEND` directive.

Action with RESTRICT

Restriction on the `MESSAGE` parameter will be ignored.

References

None.

Procedures Used

QSTOPWATCH calls the subsidiary procedure %QSTOPWATCHHLP which converts the result of the NOW function to text structures containing the date and time. QMESSAGE displays the message on screen.

Similar Procedures

None.

Example

```
CAPTION    'QSTOPWATCH example' ; STYLE=meta
QSTOPWATCH [PRINT=* ; MODE=start ; NUMBER=1 ; MESSAGE='Overall timing']
SCALAR     ntimes, nmessage ; (5,1)*20000
QSTOPWATCH [MODE=start ; TITLE='Timing of loop']
FOR [NTIMES=ntimes ; INDEX=ii]
  IF (.NOT.MODULO(ii ; nmessage))
    QSTOPWATCH [INDEX=ii ; NTIMES=ntimes]
 ENDIF
ENDFOR
QSTOPWATCH [MODE=stop]
QSTOPWATCH [NUMBER=1 ; MESSAGE='Elapsed time since program was started']
```

QTEXT

P.W. Goedhart & L.C.P. Keizer

Can be used to obtain string(s) of a text structure from screen

contents previous next

Options

TITLE = *text*	Single-valued text structure specifying the title of the screen box; must be set
MESSAGE = *text*	Single-valued text structure specifying an extra explanatory message; default *
DEFAULT = *text*	Default response; default *
NTIMES = *scalar*	The number of times an input string is prompted for; default 1
CASE = *string token*	Case to use for letters (given, lower, upper, changed, title); default given leaves the case of each letter as given on screen

Parameters

TEXT = *texts*	Text structure to save the response(s); must be set
CHECK = *scalars*	Scalar to save whether TEXT has at least one value (1) or whether it has only missing values (0)
VCHECK = *variates*	Saves a variate of the same length as the TEXT parameter with elements 1 (string present) or 0 (string missing)

Description

Procedure QTEXT can be used to obtain string(s) of the TEXT parameter from screen. This procedure uses an external WinBatch program. The NTIMES option specifies the number of strings which should be obtained from screen. If NTIMES is greater than 1, the strings of TEXT are obtained by repeatedly displaying a screen box until the text structure is fully filled, or until the Cancel button is pressed. The CHECK parameter saves whether the TEXT parameter has at least on value (1) or only missing values (0). The VCHECK parameter saves whether corresponding strings in the TEXT parameter are present (1) or not (0). In case no string is typed on screen, e.g. when just the OK button is pressed, the corresponding element of TEXT and VCHECK is set to missing and 0 respectively.

The title of the screen box can be specified by means of the TITLE option and an extra message is displayed when the MESSAGE option is set. The DEFAULT option can be used to set a default response. The CASE option can be used to change the case of the saved text. The title setting of CASE changes the case of all letters to lowercase, except the first letter which is changed to uppercase.

Method

The SUSPEND [CONTINUE=no] directive is used to invoke an external WinBatch executable.

Action with RESTRICT

The TEXT parameter is redefined in the procedure, and so restrictions on TEXT are ignored.

References

None.

Procedures Used

The BIOMETRIS procedure is used to retrieve the filename of the External WinBatch executable.

Similar Procedures

QDIRECTORY returns a directory selected by means of a directory browse dialog box on screen. QFILENAME returns a filename selected by means of a file open box on screen. QMESSAGE displays a message on screen. QPICKLIST can be used to pick one or more items from a list presented on screen. QSTOPWATCH can be used to display timing information. QYESNO can be used to choose between alternatives Yes, No and Cancel on screen.

Example

```
CAPTION   'QTEXT example' ; STYLE=meta
QTEXT     [TITLE='Example 1 of Biometris procedure QTEXT'] text
PRINT     text
QTEXT     [TITLE='Example 2 of Biometris procedure QTEXT' ; \
          MESSAGE='Input a string' ; DEFAULT='Type a string!' ; NTIMES=3] text
PRINT     text
```

QTIMEDELAY

P.W. Goedhart

Pauses execution for a specific amount of time or until a file is not opened by another application

contents previous next

Options

SECONDS = *scalar*	Number of seconds to wait; default 5.
DIRECTORY = *text*	Single-valued text which specifies the directory for the files; default *, i.e. the current working directory
PROCESSOR = *scalar*	Processor on which to run the external WinBatch executable; default * employs all available processors

Parameters

FILES = *texts*	Files for which the “open” status must be checked
OPENSTATUS = *variates*	Saves the “open” status of the files, i.e. -1 for a file that does not exist, 0 for a file that is not open by another application, and 1 for a file that is open by another application

Description

Procedure QTIMEDELAY can be used to pause execution of GenStat for a specific amount of time. When the FILES parameter is not set the SECONDS option determines how many seconds are paused. When the FILES parameter is set, execution is paused until at least one file is not open by another application. When all files are open by another application execution is paused for the specified number of seconds after which the open status of all files is determined once again. When at least one file is not open by another application the procedure is exited and the “open” status of all files can be saved by means of the OPENSTATUS parameter. The “open” status equals -1 for a file that does not exist, 0 for a file that is not open by another application, and 1 for a file that is open by another application.

The PROCESSOR option defines on which processor the external WinBatch executable must run; this might be useful for multicore processors. Note that the numbering of processors starts at zero.

Method

The SUSPEND [CONTINUE=no] directive is used to invoke an external WinBatch executable. This is either a direct call to the executable, or when PROCESSOR is specified the DOS START command with its option AFFINITY is employed.

Action with RESTRICT

Restriction on the FILES parameter are ignored.

References

None.

Procedures Used

The BIOMETRIS procedure is used to retrieve the filename of the External WinBatch executable.

Similar Procedures

None.

Example

```
CAPTION    'QTIMEDELAY example' ; STYLE=meta
QTIMEDELAY [SECONDS=10]
```

QYESNO

P.W. Goedhart & L.C.P. Keizer

Can be used to choose between alternatives Yes, No and Cancel on screen

contents previous next

Options

TITLE = *text*	Single-valued text structure specifying the title of the screen box; must be set
MESSAGE = *text*	Text structure specifying a message or question ; default *

Parameters

CHOICE = *scalar*	Saves the selected button in a scalar with the value 1 (Yes), 0 (No) or -1 (Cancel); must be set

Description

Procedure QYESNO can be used to choose between Yes, No and Cancel by means of a screen box. This procedure employs an external WinBatch program. The title of the screen box must be specified by the TITLE parameter, while the MESSAGE parameter can be used to specify an extra message or question. The special strings @hrt and @tab in the MESSAGE parameter are translated into hard returns and tabs respectively. The CHOICE parameter saves the selected button by means of a scalar with value 1 (Yes), 0 (No) and -1 (Cancel).

Method

The SUSPEND [CONTINUE=no] directive is used to invoke an external WinBatch executable.

Action with RESTRICT

The MESSAGE option can be restricted.

References

None**.**

Procedures Used

The BIOMETRIS procedure is used to retrieve the filename of the External WinBatch executable.

Similar Procedures

QDIRECTORY returns a directory selected by means of a directory browse dialog box on screen. QFILENAME returns a filename selected by means of a file open box on screen. QMESSAGE displays a message on screen. QPICKLIST can be used to pick one or more items from a list presented on screen. QSTOPWATCH can be used to display timing information. QTEXT can be used to obtain string(s) of a text structure from screen.

Example

```
CAPTION   'QYESNO example' ; STYLE=meta
TEXT      title ; 'Example of Biometris procedure QYESNO'
TEXT      message ; VALUES=!t('Do you want to continue?', \
          '@hrtYes@tabresults in choice = 1', \
          '@hrtNo@tabresults in choice = 0' , \
          '@hrtCancel@tabresults in choice = -1')
QYESNO    [TITLE=title ; MESSAGE=message] choice
PRINT     choice
```

R2NEGBINOMIAL

R.M. Harbord, R.W. Payne & P.W. Goedhart

Fits a negative binomial generalized linear model estimating the aggregation parameter

contents previous next

Options

PRINT = *string tokens*	Printed output from the analysis (model, deviance, summary, estimates, correlations, fittedvalues, accumulated, monitoring, aggregation, loglikelihood); default mode, summ, esti, aggr
AGGREGATION = *scalar*	Saves the estimate of the aggregation parameter
_2LOGLIKILEHOOD = *scalar*	Saves the value of -2 × log-likelihood
CONSTANT = *string token*	How to treat the constant (estimate, omit); default esti
FACTORIAL = *scalar*	Limit on number of factors in a treatment term; default 3
NOMESSAGE = *string tokens*	Warnings to suppress from FIT (dispersion, leverage, residual, aliasing, marginality, vertical, df, inflation, warnings); default *
FPROBABILITY = *string token*	Printing of probabilities for variance 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 (%variance, %ss, adjustedr2, r2, dispersion, %meandeviance, %deviance, aic, bic, sic); default disp
MAXCYCLE = *scalar* or *variate*	Maximum number of iteration for main, Newton-Raphson, FMINIMIZE and RCYCLE iterations; a single setting implies the same limit for all; default !(20, 20, 20, 50)
TOLERANCE = *scalar* or *variate*	Convergence criteria for deviance, aggregation, FMINIMIZE and RCYCLE; a single setting implies the same criterion for all; default !(1.0e-4, 1.0e-4, 1.0e-2, 1.0e-4)
LOWER = *scalar*	Lower limit of the aggregation parameter which is used for initial bracketing; default 0.01
UPPER = *scalar*	Upper limit of the aggregation parameter which is used for initial bracketing; default 1000

Parameters

TERMS = *formula*	List of explanatory variates and factors, or model formula (as for FIT)

Description

Procedure R2NEGBINOMIAL is a robust alternative to the RNEGBINOMIAL procedure. The latter procedure sometimes fails to estimate the aggregation parameter, further called index parameter or *k*, due to a naive starting value of the index parameter. R2NEGBINOMIAL employs the FMINIMIZE procedure to find a good starting value for *k* after which the usual Newton-Raphson method is used to solve the score equation for *k*. In addition to the options of RNEGBINOMIAL this procedure has options LOWER and UPPER which should bracket the estimate of the index parameter. Moreover the MAXCYCLE and TOLERANCE options have an extra element which is employed by the FMINIMIZE procedure. The remaining part of the description is largely copied from RNEGBINOMIAL.

The negative binomial distribution can be fitted as a generalized linear model using FIT only for a given value of the aggregation parameter *k*. R2NEGBINOMIAL extends the fitting to include estimation of *k* from the data. The negative binomial distribution is a discrete distribution with the relationship between mean and variance given by variance = mean + mean*mean/*k*, where *k* is a positive constant known as the aggregation parameter. It provides a possible model for count data that show apparent overdispersion when a Poisson model is fitted. The call to R2NEGBINOMIAL must be preceded by a MODEL statement with option DISTRIBUTION=negativebinomial and LINK= logarithm.

The AGGREGATION option allows the estimate of *k* to be saved. The _2LOGLIKELIHOOD option allows minus twice the maximized log-likelihood to be saved. This may be useful for comparing a sequence of nested models fitted by R2NEGBINOMIAL using likelihood ratio testing. Printed output is controlled by the PRINT option, which has the same settings as for the FIT directive but with the addition of aggregation to control the printing of the estimate of k and its standard error, and loglikelihood to print minus two times the log-likelihood.

The CONSTANT, FACTORIAL, NOMESSAGE, FPROBABILITY, TPROBABILITY, and SELECTION options operate in the usual way (as for example in the FIT directive). The NOMESSAGE options has an extra setting warnings which can be used to suppress warnings about non convergence of the iterative estimation procedure. The two options, MAXCYCLE and TOLERANCE, can supply variates of length 4 that are employed to control the iterative process if required. The first element of MAXCYCLE sets the maximum number of times that the model is fitted as a generalized linear model for fixed *k*, the second element sets the maximum number of Newton-Raphson iterations used to maximise the likelihood with respect to *k* for fixed fitted values, the third element sets the maximum number of iterations of the FMINIMIZE procedure which is used to obtain a good starting value for *k*, and finally the fourth element is used for setting the MAXCYCLE option of RCYCLE. The alternating cycle, controlled by the first two elements of MAXCYCLE, stops when successive values of the deviance are within a tolerance set by the first element of the TOLERANCE option and successive values of the deviance are within a tolerance set by the second element. The third element of TOLERANCE sets the convergence criterion of the FMINIMIZE procedure, while the fourth element sets the TOLERANCE option of RCYCLE.

Method

For fixed *k*, the negative binomial distribution is in the exponential family and the regression parameters determining the fitted values can be fitted as a generalized linear model using the FIT directive. For a fixed set of fitted values, *k* can be estimated by using the Newton-Raphson method to solve the score equation for *k*. Alternating between the two processes until convergence yields joint maximum likelihood estimates of *k* and the regression parameters. As the estimate of *k* is asymptotically independent of the other regression parameters, their standard errors can be obtained separately from the two processes. The standard error for *k* uses observed rather than expected information due to the use of Newton-Raphson rather than Fisher scoring.

A good starting value of *k* is obtained by the FMINIMIZE procedure which finds a minimum in the bracketing interval as specified by the LOWER and UPPER options. Note that FMINIMIZE is operating on the logarithm of *k* rather than *k* itself.

Action with RESTRICT

Any restriction applied to vectors used in the regression model applies also to the results from R2NEGBINOMIAL.

References

None.

Procedures Used

SUBSET, FMINIMIZE and %FMINIMIZE.

Similar Procedures

RNEGBINOMIAL fits the negative binomial generalised linear model.

Example

```
CAPTION   'R2NEGBINOMIAL example (see RNEGBINOMIAL)' ; STYLE=meta
FACTOR    [NVALUES=10 ; LEVELS=2 ; LABELS=!t('Continuous','Standby')] mode
READ      [PRINT=* ; SETNVALUES=yes] mode,events,time
    1  5  94.320   2  1  15.720   1  5  62.880   1 14 125.760   2  3   5.240
    1 19  31.440   2  1   1.048   2  1   1.048   2  4   2.096   2 22  10.480 :
CALCULATE logtime = LOG(time)
MODEL     [DISTRIBUTION=negativebinomial ; LINK=log ; OFFSET=logtime] events
R2NEGBIN  [PRINT=#,monitoring,loglikelihood ; NOMESSAGE=resi,disp] mode
```

RBETABINOMIAL

P.W. Goedhart

Fits the Beta-Binomial regression model to overdispersed proportions

contents previous next

Options

PRINT = *string tokens*	What to print (model, summary, estimates, correlations, monitoring); default model, summary, estimates
CONSTANT = *string token*	How to treat the constant (estimate, omit); default estimate
FACTORIAL = *scalar*	Limit for expansion of model terms; default 3
NOMESSAGE = *string tokens*	Which warning messages to suppress (aliasing, marginality, warning); default *
FULL = *string token*	Whether to assign all possible parameters to factors and interactions (yes, no); default no
RESIDUALS = *variate*	Saves the Pearson residuals
FITTEDVALUES = *variate*	Saves the fitted values
ESTIMATES = *variate*	Saves the estimates of the regression parameters
SE = *variate*	Saves the standard errors of the estimates of the regression parameters
VCOVARIANCE = *symmetric matrix*	Saves the variance-covariance matrix of the regression estimates
PHI = *scalar*	Saves the estimated overdispersion parameter φ
SEPHI = *scalar*	Saves the standard error of the estimated overdispersion parameter φ
FULLVCOVARIANCE = *symmetric matrix*	Saves the full variance-covariance matrix of all parameters, corrected for estimation of the overdispersion parameter φ
_2LOGLIKELIHOOD = *scalar*	Saves the value of -2 × log-likelihood
DF = *scalar*	Saves the residual degrees of freedom
SAVE = *pointer*	Saves additional structures
METHOD = *string token*	Algorithm for fitting the overdispersion parameter φ (Brent, mBrent, NewtonRaphson, N2ewtonRaphson, GoldenSectionSearch); default NewtonRaphson
MAXCYCLE = *scalar*	Maximum number of iterations; default 30
TOLERANCE = *scalar*	Convergence criterion; default 0.0001
INITIAL = *scalar* or *variate*	Initial values for the overdispersion parameter φ to start the iterative process; default !(0.0001, 0.01, 0.05, 0.1, 0.3, 0.5, 0.75, 0.9999)

Parameters

TERMS = *formula*	List of explanatory variates and factors, or model formula

Description

In binomial regression models, residual variability is often larger than would be expected if the data were indeed binomially distributed. This may be due to a few outliers or a poor choice of link function but often it simply indicates that the data are from a distribution more variable than the binomial. Such data are said to be "overdispersed" or to exhibit "extra-binomial variation". One way of dealing with binomial overdispersion is to assume that the data follow the Beta-Binomial distribution, see Crowder (1978) for an early reference. The Beta-Binomial distribution arises by assuming that the probability p of success follows a Beta(α, β) distribution and that conditionally on p the data are binomially(n, p) distributed. The mean and variance of the Beta-Binomial are given by $n\pi$ and $n\pi(1-\pi)(1 + \varphi(n-1))$ respectively with $\pi = \alpha/(\alpha+\beta)$ and overdispersion parameter $\varphi=1/(1+\alpha+\beta)$. The overdispersion parameter φ is limited to the interval (0,1). The response probability π is then related to the model formula employing one of the standard binomial link functions. Maximum likelihood is used to estimate the regression parameters and the overdispersion parameter φ. Note that Williams (1982) model II, implemented by the EXTRABINOMIAL procedure, is similar in spirit but uses only the first two moments of the Beta-Binomial distribution to estimate the parameters.

A call to the RBETABINOMIAL procedure must be preceded by a MODEL statement with option setting DISTRIBUTION=binomial and LINK option set to either logit, probit or complementaryloglog. The user can also choose to set the WEIGHTS, OFFSET and GROUPS options of the MODEL directive.

The options and parameter of RBETABINOMIAL are similar in many ways to the standard regression directives. There is a single parameter TERMS to define the model terms to be fitted, and the first four options, PRINT, CONSTANT, FACTORIAL, and NOMESSAGE, all have the same syntax and purpose as in FIT. The FULL option has the same purpose as in TERMS. The warning setting of NOMESSAGE suppresses warnings messages concerning the iterative fitting algorithm.

The model is fitted by an iterative process as specified by the METHOD option using either a form of Newton-Raphson (the default) or Golden Section Search. The MAXCYCLE option specifies the number of iterations, and the TOLERANCE option defines the convergence criterion. Initial values for the overdispersion parameter φ can be specified by means of the INITIAL option. Full details of the iterative procedure and the way in which the initial values are handled are given in the Method section. The iterative process can be monitored by specifying PRINT=monitoring; this will first display monitoring of the way in which an initial value for φ is obtained, followed by monitoring of the iterative process itself.

A large number of options can be used to save results from the fitted model, most of which are similar to the RKEEP directive. Fitted values and Pearson residuals can be saved by means of the FITTEDVALUES and RESIDUALS options. The ESTIMATES, SE and VCOVARIANCE options save results for the regression parameters, while PHI and SEPHI save the estimate of the overdispersion parameter φ and its standard error. Note that the standard errors thus saved are not corrected for the estimation of φ, i.e. it is assumed that the correlations between the regression parameters and φ are zero. A corrected variance-covariance matrix, see Stirling (1984), can be saved by means of the FULLVCOVARIANCE option. The value of minus twice the maximized log-likelihood and the corresponding degrees of freedom can be saved by means of the _2LOGLIKELIHOOD and DF options. This may be useful to compare nested model employing a likelihood ratio test.

Additional results can be saved by setting the SAVE option. This will save, in a pointer, (1) logit(φ); (2) the standard error of logit(φ); (3) the linear predictor; (4) the leverage; (5) the iterative weights; (6) the adjusted response; (7) the standard errors of the linear predictor; (8) the score with the first order derivative of the log-likelihood with respect to the linear predictor; (9) the contribution of each unit to the value of minus twice the maximized log-likelihood; (10) the value of the α parameter for each unit; (11) the value of the β parameter for each unit; (12) the variance of the Beta-Binomial distribution for each unit; (13) the conditional expectation of the probability given the observed value which is termed blup; (14) the number of cycles of the iterative procedure; (15) a logical value indicating non-convergence; three logical values indicating that the estimate of phi is (16) close to a limit as specified by INITIAL, (17) equal to the lower bound 0.0001, (18) equal to the upper bound 0.9999.

Method

For a fixed overdispersion parameter φ, all parameters are linear and iteratively reweighted least squares using expected information can be used to maximize the log-likelihood (Stirling, 1984). The log-likelihood only employs the LNGAMMA functions and so the first and second order derivatives are calculate by means of the DIGAMMA and TRIGAMMA function.

The overdispersion parameter φ can be estimated by the Newton-Raphson method to solve the score equation for φ (for a fixed set of fitted values). The overdispersion parameter φ is, by definition, in the interval (0,1). Therefore the score equation for logit(φ), rather than φ itself, is used. Alternating between the two processes, i.e. fixing φ and fixing the fitted values, until convergence yields joint maximum likelihood estimates of φ and the regression parameters. The iteration stops when the maximum relative change in fitted values in successive iterations is less than the tolerance, or in case the iteration number exceeds 10, when the relative change in the value of minus twice the log-likelihood is less than the tolerance divided by 1000. This all is used by the METHOD settings NewtonRaphson and N2ewtonRaphson. These two methods only differ in the way an initial value for φ is derived. The first method employs a modified version of the EXTRABINOMIAL procedure with starting value equal to the first element of INITIAL. The second method first fits the model for a number of fixed values of φ as set by the INITIAL option. This method also uses the fact that the log-likelihood as a function of φ is convex.

Alternatively the overdispersion parameter φ is estimated by a form of Golden Section Search as implemented in procedure FMINIMIZE. An initial bracketing interval is obtained by subsequently fitting the model for a number of fixed values of φ as set by the INITIAL option. See FMINIMIZE for the convergence criterion.

For most datasets METHOD=NewtonRaphson will give maximum likelihood estimates in a limited number of iterations. Occasionally, e.g. due to a flat likelihood, this estimation method fails and then METHOD=brent provides a robust alternative.

The estimate of φ is not asymptotically independent of the regression parameters, and so a corrected variance-covariance matrix is calculated by means of the method outlined in Stirling (1984). In the experience of the author of this procedure, there are only minor differences between the uncorrected and corrected asymptotic variance-covariance matrices, as saved by the VCOVARIANCE and FULLVCOVARIANCE options. The variance-covariance matrices, and standard errors, use observed rather than expected information as there is no closed form available for expected information. A confidence interval for φ can be calculated best on the logit scale, and therefore logit(φ) and its estimated standard error can be saved by means of the SAVE option.

Action with RESTRICT

Only the response variate can be restricted and the analysis is restricted accordingly.

References

Crowder, M.J. (1978). Beta-binomial anova for proportions. *Applied Statistics*, **27**, 162-167.

Stirling, W.D. (1984). Iteratively reweighted least squares for models with a linear part. *Applied Statistics*, **33**, 7-17.

Williams, D.A. (1982). Extra-binomial variation in logistic linear model. *Applied Statistics,* **31**, 144-148.

Procedures Used

FMINIMIZE, %FMINIMIZE. The subsidiary procedure %EXTRABINOMIAL implements a modified version of the EXTRABINOMIAL procedure.

Similar Procedures

EXTRABINOMIAL fits the models of Williams (1982) to overdispersed proportions. RLOGITNORMAL fits the Logistic-Normal regression model to overdispersed proportions. BBINOMIAL estimates the parameters of the beta binomial distribution for a single sample. GROBINOMIAL can be used to generate pseudo random numbers from the beta-binomial distribution.

Example

```
CAPTION   'RBETABINOMIAL example',\
          !t('A 2 x 2 factorial experiment comparing germination',\
          'of two types of seed and two root extracts (Crowder, M.J.,',\
          '1978, Applied Statistics, 27, 34-37).') ; STYLE=meta,plain
FACTOR    [LABELS=!T(O_75,O_73); VALUES=1,10(1,2)] Seed
FACTOR    [LABELS=!T(Bean,Cucumber); VALUES=5(1,2),2,5(1,2)] RtExtrct
VARIATE   NGerm,NSeeds ; VALUES=\
          !(10,23,23,26,17,5,53,55,32,46,10, 8,10, 8,23,0, 3,22,15,32,3), \
          !(39,62,81,51,39,6,74,72,51,79,13,16,30,28,45,4,12,41,30,51,7)
MODEL     [DISTRIBUTION=binomial; LINK=logit] NGerm; NBINOMIAL=NSeeds
RBETABINO [PRINT=#,monitoring] Seed*RtExtrct
RBETABINO [PRINT=#,monitoring ; METHOD=brent] Seed*RtExtrct

CAPTION   'Show equivalence with BBINOMIAL for a single sample' ; STYLE=meta
RBETABINO [PRINT=esti ; ESTIMATES=esti1 ; SE=se1 ; PHI=phi1 ; SEPHI=sephi1]
BBINOMIAL NGerm ; NBINOMIAL=NSeeds ; MU=mu ; THETA=theta ; \
          SEMU=sem ; SETHETA=seth
CALCULATE esti2 = LOGIT(100*mu)
CALCULATE phi2  = theta/(1+theta)
CALCULATE se2 = sem/(mu*(1-mu))
CALCULATE sephi2 = seth/(1+theta)
PRINT     [RLPRINT=*] esti1,esti2, phi1,phi2 ; DECIMALS=6
PRINT     [RLPRINT=*] se1,se2, sephi1,sephi2 ; DECIMALS=6
```

RENAMEPOINTER

L.C.P. Keizer & J.T.N.M. Thissen

Renames the structures of a pointer

contents previous next

Options

`SCOPE` = *string token*	This allows 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`); default `global`

Parameters

`POINTER` = *pointers*	Pointer whose structures are to be renamed; must be set
`NAME` = *texts*	Text structure with new names for the structures of `POINTER`; must be set

Description

Procedure `RENAMEPOINTER` can be used to rename the structures of a pointer. This is especially useful when new names are read from file. The pointer and new names must be specified by means of the `POINTER` and `NAME` parameters. The length of `NAME` should be equal to the number of structures of `POINTER`. The `SCOPE` option is similar to that of the `ASSIGN` directive.

Method

Directive `ASSIGN`, with the `SCOPE` option set, is printed to a text structure and then executed.

Action with RESTRICT

Restrictions on the `NAME` parameter are ignored.

References

None.

Procedures Used

None.

Similar Procedures

`FPOINTER` forms a pointer from a text structure.

Example

```
CAPTION       'RENAMEPOINTER example' ; STYLE=meta
VARIATE       [VALUES=1...4] x[1] ; DECIMALS=0
VARIATE       [VALUES=5...8] x[2] ; DECIMALS=0
TEXT          [VALUES=A, B, C, D] x[3]
FACTOR        [LEVELS=2 ; VALUES=2(1,2)] x[4] ; DECIMALS=0
TEXT          [VALUES=variate1, variate2, text, factor] newname
RENAMEPOINTER POINTER=x ; NAME=newname
PRINT         x[] ; FIELD=12
```

RGDISPLAY

P.W. Goedhart

Displays and stores the non-groups parameters of a within-groups regression

contents previous next

Options

PRINT = *string token*	Printed output required (estimates); default estimates

Parameters

ESTIMATES = *variate*	To save estimates of the non-groups parameters of a within-groups regression
SE = *variate*	To save standard errors of the estimates of the non-groups parameters
VCOVARIANCE = *symmetric matrix*	To save the variance-covariance matrix of the estimates of the non-groups parameters
LABELS = *text*	To save labels of the estimates of the non-groups parameters
EGROUPS = *variate*	To save estimates of the groups parameters
SEGROUPS = *variate*	To save standard errors of the estimates of the groups parameters

Description

A within-groups regression model can be fitted by specifying the GROUPS option in the MODEL directive. Use of GROUPS gives less information than you would get if you included the grouping factor explicitly in the model, because leverages, predictions and some parameter correlations are not formed, but it saves space and time in fitting the model when the groups factor has many levels. After such a model is fitted, all the estimated groups parameters are standardly displayed and stored. However, these parameters are frequently of less interest. Procedure RGDISPLAY can be used to display and store the non-groups parameters only. The estimates, standard errors, variance-covariance matrix and labels can be saved by means of parameters ESTIMATES, SE, VCOVARIANCE and LABELS. The estimates of the groups parameters and their standard errors can be saved by means of EGROUPS and SEGROUPS. The PRINT option can be used to suppress printing of the estimates.

Method

The information about the GROUPS option of the MODEL directive is retrieved by means of the OMODEL option of RKEEP. The LABELS are retrieved from the special regression save structure.

Action with RESTRICT

Not relevant.

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION   'RGDISPLAY example', !t('Trend analysis of Skylark counts', \
          'in the Netherlands'), ' ' ; STYLE=meta, 2(plain)
UNIT      [202]
FACTOR    site, time ; DECIMALS=0
FACTOR    [LABELS=!t(Dunes, Heath)] habitat
READ      count, site, time, habitat
   11  1 1 2   8  1 2 2   5  1 3 2   4  1 4 2  10  1 5 2   7  1 6 2  15  2 1 2
    9  2 2 2   7  2 3 2   8  2 4 2   6  2 5 2  21  2 6 2  41  3 1 2  29  3 2 2
   36  3 3 2  37  3 4 2  49  3 5 2  74  3 6 2  68  3 7 2  97  3 8 2  13  4 1 2
   11  4 2 2   6  4 3 2  11  4 4 2  17  4 5 2  22  4 6 2  28  4 7 2  36  4 8 2
    1  5 3 1   1  5 6 1  15  6 1 1  16  6 2 1  14  6 3 1  12  6 4 1  12  6 5 1
   13  6 6 1  12  6 7 1  11  6 8 1  11  7 1 2   9  7 2 2   7  7 3 2   4  7 4 2
    5  7 5 2   7  7 6 2  10  7 7 2  10  7 8 2   1  8 1 1   4  9 1 1   1 10 1 2
   23 11 1 2  16 11 2 2  40 11 3 2  35 11 4 2  28 11 5 2  21 11 6 2  31 11 7 2
   18 11 8 2  25 12 1 2  12 12 2 2  11 12 3 2  12 12 4 2  15 12 5 2  17 12 6 2
    2 13 3 2   2 13 4 2   1 13 6 2   1 13 7 2   1 13 8 2   4 14 1 2   4 14 2 2
    4 14 3 2   7 14 4 2   6 14 5 2   8 14 6 2   8 14 7 2   8 14 8 2   7 15 1 2
    5 15 3 2   4 15 4 2   5 15 5 2   1 16 3 2   1 16 4 2   1 16 5 2   2 17 1 1
    1 17 4 1   1 18 1 1   1 19 1 1   5 20 1 1   3 20 2 1   3 21 1 1   3 21 2 1
    2 21 3 1   8 22 1 1   3 22 2 1   1 22 3 1   1 22 5 1   3 23 1 1   1 23 2 1
    3 23 6 1   3 24 1 1   3 24 2 1   5 24 3 1   3 24 4 1   2 24 5 1   3 24 6 1
    2 24 7 1   1 24 8 1   6 25 1 1   8 25 2 1   4 25 3 1   2 25 4 1   2 25 5 1
    1 25 6 1   1 25 8 1   1 26 3 1   4 27 2 2   7 27 3 2   6 27 4 2   7 27 5 2
    7 27 6 2   7 27 7 2  14 28 2 2   1 29 7 1   1 30 1 1   1 31 2 1   2 31 3 1
    1 31 4 1   1 31 5 1   2 32 3 2   1 32 4 2   2 32 5 2   1 32 6 2   3 32 8 2
    4 33 2 2   4 33 3 2   6 33 4 2   7 33 5 2   5 33 6 2   3 33 7 2   1 34 3 1
    1 35 3 2   2 35 4 2   2 35 5 2   2 35 6 2   3 35 7 2   3 35 8 2   1 36 3 1
    1 36 4 1  57 37 3 2  60 37 4 2  55 37 5 2  50 37 6 2  44 37 7 2   7 38 3 1
    8 38 4 1   5 38 5 1  10 38 6 1   5 38 7 1   6 38 8 1   1 39 4 1  88 40 3 2
   80 40 4 2 108 40 5 2 104 40 6 2 131 40 7 2 113 40 8 2   6 41 4 1   3 41 5 1
    4 41 6 1   1 42 5 1   1 42 7 1   1 43 5 2   8 44 6 2  16 45 4 2   3 46 4 1
    4 46 5 1   2 46 6 1   1 46 7 1   2 46 8 1   7 47 1 1   4 47 2 1   4 47 3 1
    3 47 4 1   1 47 5 1   2 47 6 1   4 47 7 1   4 47 8 1   1 48 7 2   2 49 1 2
    2 49 7 2   2 49 8 2  20 50 6 1   1 51 6 1   9 52 6 1  10 52 7 1   8 52 8 1
    1 53 7 2  16 54 8 2   1 55 3 1   1 55 4 1   1 55 5 1   1 55 6 1 :
CALCULATE timelin, timequad = time * (1, time)
MODEL     [DISTRIBUTION=poisson ; GROUPS=site] count
TERMS     (time + timelin + timequad)*habitat
SETOPTION [DIRECTIVE=ADD] NOMESSAGE ; !t(aliasing)
FIT       [PRINT=*] timelin
ADD       [PRINT=*] timequad
ADD       [PRINT=*] time
ADD       [PRINT=*] habitat
ADD       [PRINT=*] timelin/habitat
ADD       [PRINT=*] timequad/habitat
ADD       [PRINT=*] time/habitat
RDISPLAY  [PRINT=accumulated ; FPROBABILITY=yes]
FIT       [NOMESSAGE=residual,leverage] timelin/habitat + timequad
RGDISPLAY
```

RLMS

P.W. Goedhart

Does regression by least median squares

contents previous next

Options

PRINT = *string tokens*	Printed output required (model, estimates, fittedvalues); default model, estimates
CONSTANT = *string token*	How to treat constant (estimate, omit); default estimate
INDEXPLOT = *string tokens*	What to display in an indexplot (residuals, diagnostics); default *
ALGORITHM = *string token*	Which algorithm to use (extensive, quick); default extensive
NOMESSAGE = *string token*	Which warning messages to suppress (residuals); default *

Parameters

X = *pointers*	Pointer containing the predictor variables to enter the LMS regression
RESIDUALS = *variates*	To save the residuals of the LMS regression
FITTEDVALUES = *variates*	To save the fitted values of the LMS regression
DIAGNOSTICS = *variates*	To save the resistant diagnostics of the LMS regression
ESTIMATES = *variates*	To save the estimated parameters of the LMS regression
SCALE = *scalars*	To save the scale estimate of the LMS regression

Description

Classical least squares regression (LS) consists of minimizing the sum of squared residuals. Outliers, both in the response variable and in the predictor variables, may have a strong influence on the least squares estimates. Outliers can be identified by diagnostic techniques using residuals and leverages. However, even careful residual analysis may not always reveal (multiple) outliers. Moreover, the successful use of diagnostic procedures often requires abilities beyond those of a naive user of regression techniques.

To remedy these problems, robust regression methods have been developed that are not so easily affected by outliers. Observations far away from the robust fit are then identified by their large residuals from it. One such technique is least median squares (LMS) in which the *median* of the squared residuals is minimized, see Rousseeuw (1984) or Rousseeuw and Leroy (1987). In the special case of simple linear regression, LMS corresponds to finding the narrowest strip covering half of the observations.

A call to RLMS must be preceded by a MODEL statement in which the response variable is specified. Only the first response variable is analysed and the WEIGHTS, OFFSET and GROUPS options of MODEL are ignored for the LMS fit. Generalized linear models are not allowed. Parameter X specifies the set of predictors to enter the regression. The CONSTANT option controls whether the constant parameter should be included in the model. Residuals, fitted values and estimates of the least median squares regression can be saved using parameters RESIDUALS, FITTEDVALUES and ESTIMATES. DIAGNOSTICS saves the resistant diagnostic which aims at identifying all points that are either outliers or large leverage points. A resistant diagnostic exceeding 2.5 is considered to be large. SCALE saves a robust estimate of scale which is essentially the residual standard deviation of the units with small residuals. The resistant diagnostic and the scale estimate are fully defined in Rousseeuw and Leroy (1987, page 238 and 202). RKEEP can be used to store results of the ordinary least squares regression which is also performed in the procedure.

Output is controlled by the PRINT and INDEXPLOT options. The model and estimates settings of PRINT are default and give a description of the model and estimates of both the LMS and the ordinary LS regression along with estimates of scale. If the LS fit agrees closely with the LMS fit, the LS result can be trusted. The fittedvalues setting of PRINT gives a table of unit labels, response variate, fitted values, residuals (scaled by the estimate of scale) and resistant diagnostics. Option INDEXPLOT provides indexplots of residuals and diagnostics, which are displayed in the first graphical window using the first pen. The NOMESSAGE option controls printing of warning messages of units with residual larger than 2.0 or diagnostic larger than 2.5. No warnings are printed when PRINT is set to fittedvalues.

Computation of LMS is similar in spirit to the bootstrap. The `ALGORITHM` option controls the number of subsamples to be drawn for a given data set. In general the setting `extensive` is advised.

Method

The procedure passes the problem to a C# .NET Framework 4 executable by using the `SUSPEND` directive. The C# program is a translation of the Fortran program PROGRESS (Leroy and Rousseeuw, 1984). The algorithm is similar in spirit to the bootstrap. With *p* explanatory variables, it repeatedly draws subsamples of *p* different observations, determines the exact fitting regression surface through the *p* points and calculates the median of the squared residuals with respect to the whole data set. The subsample with minimal median is retained. This implies that different LMS estimates can be obtained by changing the order of the observations.The number of subsamples depends on the number of observations, the number of regressors and the setting of option `ALGORITHM` (Rousseeuw and Leroy, 1987).

Action with RESTRICT

Only the response variate can be restricted an the analysis is restricted accordingly. Parameters `RESIDUALS`, `FITTEDVALUES` and `DIAGNOSTICS` are not restricted on output.

References

Leroy, A. and Rousseeuw, P. (1984). *PROGRESS: A program for robust regression*. Technical report 201, Center for Statistics and O.R., University of Brussels, Belgium.

Rousseeuw, P.J. (1984). Least median squares regression. *Journal of the American Statistical Association*, **79**, 871-880.

Rousseeuw, P.J. and Leroy, A.M. (1987). *Robust regression and outlier detection*. Wiley. New York.

Procedures Used

The `BIOMETRIS` procedure is used to retrieve the filename of the external C# executable.

Similar Procedures

None.

Example

```
CAPTION  'RLMS example: Analysis of the Stackloss Data.', \
         !t('Most statisticians that analysed these data concluded that', \
         'observations 1,3,4, and 21 are outliers and that observation', \
         '2 might be an outlier. The two calls to LMS illustrate the', \
         'possible differences between LMS estimates when different sets', \
         'of random subsamples are considered.'), \
         !t('Comparison of the LS residuals and the LMS residuals reveals', \
         'that LMS immediately points at observations 1,2,3,4 and 21,', \
         'while the LS fit only points at 17 having a large leverage', \
         'and 21 having a large residual.'), ' ' ; STYLE=meta, 3(plain)
UNIT     [21]
READ     Unit, Rate,Temp,Acid,Stackloss
  1 80 27 89 42  2 80 27 88 37  3 75 25 90 37  4 62 24 87 28  5 62 22 87 18
  6 62 23 87 18  7 62 24 93 19  8 62 24 93 20  9 58 23 87 15 10 58 18 80 14
 11 58 18 89 14 12 58 17 88 13 13 58 18 82 11 14 58 19 93 12 15 50 18 89  8
 16 50 18 86  7 17 50 19 72  8 18 50 19 79  8 19 50 20 80  9 20 56 20 82 15
 21 70 20 91 15 :
MODEL    Stackloss
RLMS     [ALGORITHM=quick] X=!P(Rate,Temp,Acid)
RLMS     [ALGORITHM=extensive ; INDEXPLOT=residual] X=!P(Rate,Temp,Acid) ; \
         RESIDUALS=reslms ; DIAGNOSTIC=diaglms
RKEEP    RESIDUALS=residu ; LEVERAGES=leverage
PRINT    Unit, reslms, diaglms, residu, leverage ; DECIMALS=0,2,2,2,3
```

RLOGITNORMAL

P.W. Goedhart

Fits the Logistic-Normal regression model to overdispersed proportions

contents previous next

Options

`PRINT` = *string tokens*	What to print (`model`, `summary`, `estimates`, `correlations`, `monitoring`); default `model`, `summary`, `estimates`
`CONSTANT` = *string token*	How to treat the constant (`estimate`, `omit`); default `estimate`
`FACTORIAL` = *scalar*	Limit for expansion of model terms; default 3
`NOMESSAGE` = *string tokens*	Which warning messages to suppress (`aliasing`, `marginality`, `warning`); default *
`FULL` = *string token*	Whether to assign all possible parameters to factors and interactions (`yes`, `no`); default `no`
`RESIDUALS` = *variate*	Saves the leverage corrected Pearson residuals
`FITTEDVALUES` = *variate*	Saves the fitted values
`ESTIMATES` = *variate*	Saves the estimates of the regression parameters
`SE` = *variate*	Saves the standard errors of the estimates of the regression parameters
`VCOVARIANCE` = *symmetric matrix*	Saves the variance-covariance matrix of the regression estimates
`PHI` = *scalar*	Saves the estimated overdispersion parameter φ
`SEPHI` = *scalar*	Saves the standard error of the estimated overdispersion parameter φ
`FULLVCOVARIANCE` = *symmetric matrix*	Saves the full variance-covariance matrix of all parameters, i.e. the regression parameters and the dispersion parameter
`_2LOGLIKELIHOOD` = *scalar*	Saves the value of -2 × log-likelihood
`DF` = *scalar*	Saves the residual degrees of freedom
`SAVE` = *pointer*	Saves additional structures
`METHOD` = *string token*	Algorithm for fitting the overdispersion parameter φ (`Brent`, `mBrent`, `GoldenSectionSearch`, `NewtonRaphson`, `N2ewtonRaphson`); default `NewtonRaphson`
`NPOINTS` = *scalar*	Number of integration points for Gauss-Hermite integration; default 32
`MAXCYCLE` = *scalar*	Maximum number of iterations; default 30
`TOLERANCE` = *scalar*	Convergence criterion; default 0.0001
`INITIAL` = *scalar* or *variate*	Initial values for the overdispersion parameter φ to start the iterative process; default !(0.0001, 0.01, 0.1, 1, 10, 100, 1000)

Parameters

`TERMS` = *formula*	List of explanatory variates and factors, or model formula

Description

In binomial regression models, residual variability is often larger than would be expected if the data were indeed binomially distributed. This may be due to a few outliers or a poor choice of link function but often it simply indicates that the data are from a distribution more variable than the binomial. Such data are said to be "overdispersed" or to exhibit "extra-binomial variation". One way of dealing with binomial overdispersion is to assume that there is an extra normally distributed random effect, with mean zero and variance φ, on the scale of the linear predictor. The resulting distribution is sometimes termed Logit-Normal. The `RLOGITNORMAL` procedure estimates the parameters of this model using Maximum Likelihood. The likelihood cannot be calculated analytically and therefor Gauss-Hermite integration is used. This is an approximate method which is sometimes inaccurate without giving notice, see Lesaffre & Spiessens (2001). Note that Williams (1982) model III, implemented by the `EXTRABINOMIAL` procedure, fits the same model using only the first two moments of the Logit-Normal distribution to estimate the parameters.

A call to the RLOGITNORMAL procedure must be preceded by a MODEL statement with option setting DISTRIBUTION=binomial and LINK option set to either logit, probit or complementaryloglog. The user can also choose to set the WEIGHTS, OFFSET and GROUPS options of the MODEL directive.

The options and parameter of RLOGITNORMAL are similar in many ways to the standard regression directives. There is a single parameter TERMS to define the model terms to be fitted, and the first four options, PRINT, CONSTANT, FACTORIAL, and NOMESSAGE, all have the same syntax and purpose as in FIT. The FULL option has the same purpose as in TERMS. The warning setting of NOMESSAGE suppresses warnings messages concerning the iterative fitting algorithm.

The number of Gauss-Hermite integration points can be specified by the NPOINTS option. It is advised to use sufficient integration points especially when the dispersion parameter is large. The model is fitted by an iterative process as specified by the METHOD option using either a form of Newton-Raphson (the default) or a form of Golden Section Search. The MAXCYCLE option specifies the number of iterations, and the TOLERANCE option defines the convergence criterion. Initial values for the overdispersion parameter φ can be specified by means of the INITIAL option. Full details of the iterative procedure and the way in which the initial values are handled are given in the Method section. The iterative process can be monitored by specifying PRINT=monitoring; this will first display monitoring of the way in which an initial value for φ is obtained, followed by monitoring of the iterative process itself.

A large number of options can be used to save results from the fitted model, most of which are similar to the RKEEP directive. Fitted values and leverage corrected Pearson residuals can be saved by means of the FITTEDVALUES and RESIDUALS options. The ESTIMATES, SE and VCOVARIANCE options save results for the regression parameters, while PHI and SEPHI save the estimate of the overdispersion parameter φ and its standard error. Note that all standard errors thus saved are corrected for the estimation of φ. The full covariance matrix with covariances between the regression parameters and the dispersion parameter can be saved by means of the FULLVCOVARIANCE option. The value of minus twice the maximized log-likelihood and the corresponding degrees of freedom can be saved by means of the _2LOGLIKELIHOOD and DF options. This may be useful to compare nested model employing a likelihood ratio test.

Additional results can be saved by setting the SAVE option. This will save, in a pointer, (1) log(φ); (2) the standard error of log(φ); (3) the linear predictor; (4) the leverage; (5) the iterative weights which includes the regression weights; (6) the adjusted response; (7) the standard errors of the linear predictor; (8) the score with the first order derivative of the log-likelihood with respect to the linear predictor; (9) the contribution of each unit to the value of minus twice the maximized log-likelihood; (10) the variance of the Logit-Normal distribution for every value of the linear predictor; (11) the conditional expectation of the probability given the observed value which is termed blup; (12) the number of cycles of the iterative procedure; (13) a logical value indicating non-convergence; three logical values indicating that the estimate of φ is (14) close to a limit as specified by INITIAL, (15) equal to the lower bound 0.0001, (16) equal to the upper bound 1000.

Method

For a fixed overdispersion parameter φ, all parameters are linear and iteratively reweighted least squares (IRLS) can be used to maximize the log-likelihood (Stirling, 1984). The log-likelihood is intractable and Gauss-Hermite integration is used to approximate it.

The overdispersion parameter φ can be estimated by the Newton-Raphson method to solve the score equation for φ (for a fixed set of fitted values). The overdispersion parameter φ is, by definition, positive and therefore the score equation for log(φ), rather than φ itself, is used. Alternating between the two processes, i.e. IRLS and Newton-Raphson, until convergence yields joint maximum likelihood estimates of φ and the regression parameters. The iteration stops when the maximum relative change in likelihood contributions in successive iterations is less than the tolerance, or in case the iteration number exceeds 10, when the relative change in the value of minus twice the log-likelihood is less than the tolerance divided by 1000. This all is used by the METHOD settings NewtonRaphson and N2ewtonRaphson. These two methods only differ in the way an initial value for φ is derived. The first method employs a modified version of the EXTRABINOMIAL procedure with starting value equal to the first element of INITIAL. The second method first fits the model for a number of fixed values of φ as set by the INITIAL option. This last method also uses the fact that the log-likelihood as a function of φ is convex.

Alternatively the overdispersion parameter φ is estimated by a form of Golden Section Search as implemented by the FMINIMIZE procedure. An initial bracketing interval is obtained by subsequently fitting the model for a number of fixed values of φ as set by the INITIAL option. See FMINIMIZE for the convergence criterion.

For most datasets METHOD=NewtonRaphson will give maximum likelihood estimates in a limited number of iterations. Occasionally, e.g. due to a flat likelihood, this estimation method fails and then METHOD=brent provides a robust alternative.

The estimate of φ is not asymptotically independent of the regression parameters, and so a corrected variance-covariance matrix is calculated by means of the method outlined in Stirling (1984). The variance-covariance matrices, and standard errors, use observed rather than expected information as there is no closed form available for expected information. A confidence interval for φ can be calculated best on the log scale, and therefore log(φ) and its estimated standard error can be saved by means of the SAVE option. Note that the standard errors of the linear predictor are saved from the last IRLS fit and are thus **not** corrected for the estimation of φ.

Action with RESTRICT

Only the response variate can be restricted and the analysis is restricted accordingly.

References

Lesaffre, E. & Spiessens, B. (2001). On the effect of the number of quadrature points in a logistic random-effects model: an example. *Applied Statistics*, **50**, 325-335.

Stirling, W.D. (1984). Iteratively reweighted least squares for models with a linear part. *Applied Statistics*, **33**, 7-17.

Williams, D.A. (1982). Extra-binomial variation in logistic linear model. *Applied Statistics,* **31**, 144-148.

Procedures Used

FMINIMIZE, %FMINIMIZE, GAUSPOINTS. The subsidiary procedure %EXTRABINOMIAL implements a modified version of the EXTRABINOMIAL procedure.

Similar Procedures

EXTRABINOMIAL fits the models of Williams (1982) to overdispersed proportions. RBETABINOMIAL fits the beta-binomial regression model. GROBINOMIAL can be used to generate pseudo random numbers from the logit-normal distribution.

Example

```
CAPTION   'RLOGITNORMAL example',\
          !t('A 2 x 2 factorial experiment comparing germination',\
          'of two types of seed and two root extracts (Crowder, M.J.,',\
          '1978, Applied Statistics, 27, 34-37).') ; STYLE=meta,plain
FACTOR    [LABELS=!T(O_75,O_73); VALUES=1,10(1,2)] Seed
FACTOR    [LABELS=!T(Bean,Cucumber); VALUES=5(1,2),2,5(1,2)] RtExtrct
VARIATE   NGerm,NSeeds ; VALUES=\
          !(10,23,23,26,17,5,53,55,32,46,10, 8,10, 8,23,0, 3,22,15,32,3), \
          !(39,62,81,51,39,6,74,72,51,79,13,16,30,28,45,4,12,41,30,51,7)
MODEL     [DISTRIBUTION=binomial; LINK=logit] NGerm; NBINOMIAL=NSeeds
RLOGITNORMAL [PRINT=#,monitoring] Seed*RtExtrct
RLOGITNORMAL [PRINT=#,monitoring ; METHOD=brent] Seed*RtExtrct
```

RMLSTACK

P.W. Goedhart

Stacks data to allow the fitting of a multinomial logistic regression model

contents previous next

Options

`OLDRESPONSE` = *factor* or *variates*	Specifies the multinomial response, either as a list of variates or as a single factor for presence/absence data; must be set.
`NEWRESPONSE` = *variate*	Saves the multinomial response as a stacked variate; must be set
`GROUPS` = *factor*	Saves a stacked factor which corresponds to the original units; must be set
`CATEGORY` = *factor*	Saves a stacked factor which corresponds to the multinomial categories; must be set
`IDUNITS` = *variate* or *text*	Specifies the labels of the `GROUPS` factor

Parameters

`OLDVECTORS` = *variates* and/or *factors*	List of explanatory variates and/or factors which will be used in the multinomial logistic regression model
`NEWVECTORS` = *variates* and/or *factors*	Saves the stacked explanatory variates and/or factors

Description

The multinomial logistic regression model, see e.g. section 5.2.4 of McCullagh and Nelder (1989), can be fitted by employing a relation between the multinomial and Poisson likelihood. Suppose the multinomial responses are available as a set of variates, one variate for each category. Suppose further that there are a number of explanatory variates and/or factors. To fit the multinomial logistic regression model in the GLM Poisson framework, the set of response variates must first be stacked into a single variate. The explanatory vectors must also be stacked by repeating their values. Moreover a factor is needed which corresponds to the multinomial categories of the stacked response, as well as a groups factor defining the original units. The parameters of the multinomial logistic model can then be obtained by fitting interactions between the category factor and the explanatory vectors. The fitted model must contain in addition a parameter for each original unit, which can be handled by employing the `GROUPS` option of the `MODEL` directive.

The multinomial response must be specified by means of the `OLDRESPONSE` option. If `OLDRESPONSE` is set to a list of variates, `NEWRESPONSE` saves the stacked variates. Alternatively when `OLDRESPONSE` is set to a single factor, a set of variates is created internally, one for each factor level. Each variate contains presence (1) or absence (0) of the corresponding factor level, and `NEWRESPONSE` saves the stacked variates. The groups factor and multinomial categories factor must be saved by means of the options `GROUPS` and `CATEGORY`. The `IDUNITS` option can be set to specify the labels of the `GROUPS` factor. Labels of the `CATEGORY` factor are set to the names of the `OLDRESPONSE` variates or are duplicated from the `OLDRESPONSE` factor. The original explanatory vectors, which must all be of the same length as the `OLDRESPONSE` parameter, are specified by the `OLDVECTORS` parameter and identifiers for the vectors to contain the stacked vectors are specified by the `NEWVECTORS` parameter. If `NEWVECTORS` is not set, the `OLDVECTORS` vectors are redefined to store the stacked vectors instead of their original values. The multinomial logistic regression model can then be fitted as follows

```
RMLSTACK [OLDRESPONSE=y[1...3] ; NEWRESPONSE=ystack ; \
         GROUPS=group ; CATEGORY=cat] variate, factor ; vstack, fstack
MODEL    [DISTRIBUTION=poisson ; GROUPS=group] ystack
FIT      [NOMESSAGE=alias] cat/(fstack*vstack)
```

Method

The procedure uses standard GenStat directives for data manipulation.

Action with RESTRICT

The OLDRESPONSE variates or factor can be restricted. The number of values of the NEWRESPONSE variate is as if the OLDRESPONSE structures have not been restricted, but the NEWRESPONSE variate has missing values for units excluded by the restriction. Restrictions on the OLDVECTORS vectors are ignored.

References

McCullagh, P. and Nelder, J.A. (1989). *Generalized linear models, second edition*. Chapman and Hall, London.

Procedures Used

None.

Similar Procedures

Procedure RMLUNSTACK “unstacks” fittedvalues of a multinomial logistic regression model. Procedure RGDISPLAY displays and stores the non-groups parameters of a within-groups regression. Procedure STACK combines several data sets by "stacking" the corresponding vectors and procedure UNSTACK splits vectors into individual vectors according to levels of a factor.

Example

```
CAPTION   'RMLSTACK example', !t('Data taken from page 179 of McCullagh', \
          'and Nelder (1989). Generalized linear models, second edition.', \
          'Chapman and Hall.'), ' ' ; STYLE=meta,2(plain)
UNIT      [8]
READ      [PRINT=*] x, respons[1...3]
     5.8  98  0  0      15.0  51  2  1
    21.5  34  6  3      27.5  35  5  8
    33.5  32 10  9      39.5  23  7  8
    46.0  12  6 10      51.5   4  2  5  :
CALCULATE logx = log(x)
RMLSTACK  [OLDRESPONSE=respons[] ; NEWRESPONSE=y ; GROUPS=groups ; \
          CATEGORY=cat] logx ; logxstack
MODEL     [DISTRIBUTION=poisson ; GROUPS=groups] y
TERMS     cat/logxstack
SETOPTION [DIRECTIVE=ADD] NOMESSAGE ; !t(aliasing)
FIT       [PRINT=*] cat
ADD       [PRINT=accu,esti ; FPROBABILITY=yes] cat.logxstack
CAPTION   'A common slope for categories 2 and 3 can also be fitted:' ; \
          STYLE=meta
VARIATE   common ; (cat.IN.!(2,3)) * logxstack
TERMS     cat/logxstack + common
FIT       [PRINT=*] cat
ADD       [PRINT=* ; FPROBABILITY=yes] common
ADD       [PRINT=accu ; FPROBABILITY=yes] cat.logxstack
```

RMLUNSTACK

P.W. Goedhart

Unstacks results of a multinomial logistic regression model

contents previous next

Options

GROUPS = *factor*	Specifies the stacked groups factor which corresponds to the original units of the multinomial response; must be set
CATEGORY = *factor*	Specifies the stacked factor which corresponds to the multinomial categories; must be set
CONDITION = *expression*	Logical expression to define which units are to be included

Parameters

RESULTS = *variates*	Specifies results (fittedvalues, residuals, leverages) of a multinomial logistic regression model
SRESULTS = *pointers*	Saves results for each multinomial category
SPROBABILITIES = *pointers*	Saves fitted probabilities for each multinomial category; only useful when RESULTS is set to fittedvalues

Description

Procedure RMLSTACK can be used to prepare multinomial data for the fitting of a multinomial logistic regression model. RMLUNSTACK can be used to convert the stacked results, such as fittedvalues or residuals, of such a model to a pointer of variates, one variate with results for each multinomial category. In addition, for fittedvalues, a pointer of fitted probabilities can be saved.

The GROUPS and CATEGORY options settings must be identical to those used in the RMLSTACK procedure. The results of the multinomial logistic regression model must be specified by the RESULTS parameter, and pointers to “unstacked” results and probabilities can be saved by means of SRESULTS and SPROBABILITIES. The CONDITION option can be used to save a limited number of units. The CONDITION expression must yield a variate with number of values equal to the number of levels of the GROUPS factor.

Method

The procedure uses standard GenStat directives for data manipulation. The fitted probabilities in the pointer SPROBABILITIES are obtained by dividing SRESULTS[] by the sum of SRESULTS[].

Action with RESTRICT

The GROUPS and CATEGORY options must not be restricted. Restrictions on the RESULTS parameter are ignored.

References

None.

Procedures Used

None.

Similar Procedures

Procedure RMLSTACK “stacks” data to allow the fitting of a multinomial logistic regression model. Procedure STACK combines several data sets by "stacking" the corresponding vectors and procedure UNSTACK splits vectors into individual vectors according to levels of a factor.

Example

```
CAPTION   'RMLUNSTACK example', !t('Data taken from page 179 of McCullagh', \
          'and Nelder (1989). Generalized linear models, second edition.', \
          'Chapman and Hall.'), ' ' ; STYLE=meta,2(plain)
UNIT      [8]
READ      [PRINT=*] x, respons[1...3]
   5.8  98  0  0      15.0  51  2  1    21.5  34  6  3      27.5  35  5  8
  33.5  32 10  9      39.5  23  7  8    46.0  12  6 10      51.5   4  2  5 :
CALCULATE logx = log(x)
RMLSTACK  [OLDRESPONSE=respons[] ; NEWRESPONSE=y ; GROUPS=groups ; \
          CATEGORY=cat] logx ; logxstack
MODEL     [DISTRIBUTION=poisson ; GROUPS=groups] y
TERMS     cat/logxstack
FIT       [PRINT=* ; NOMESSAGE=alias] cat
ADD       [NOMESSAGE=alias] cat.logxstack
RKEEP     FITTEDVALUES=fitted ; RESIDUAL=residual
RMLUNSTACK [GROUPS=groups ; CATEGORY=cat] RESULTS=fitted,residual ; \
          SRESULTS=fit,res ; SPROBABILITIES=prob,*
PRINT     x, fit[], prob[], res[] ; FIELD=8 ; DECIMALS=1,3(2,3,2)
CAPTION   'RMLUNSTACK can also be used to obtain predictions:' ; STYLE=meta
VARIATE   xpredict ; !(5,10...60)
CALCULATE logxpredict = LOG(xpredict)
SCALAR    missing
STACK     xp, yp[1...3] ; V1=logx,respons[] ; V2=logxpredict,3(missing)
RMLSTACK  [OLDRESPONSE=yp[] ; NEWRESPONSE=y ; GROUPS=groups ; \
          CATEGORY=cat] xp ; xpstack
MODEL     [DISTRIBUTION=poisson ; GROUPS=groups] y
FIT       [PRINT=summary ; NOMESSAGE=alias] cat/xpstack
RKEEP     FITTED=fitted
RMLUNSTACK [GROUPS=groups ; CATEGORY=cat ; CONDITION=yp[1].EQ.missing] \
          RESULTS=fitted ; SPROBABILITIES=prob
PRINT     xpredict, prob[] ; FIELD=10 ; DECIMALS=0,3(3)
```

RPLS

P.W. Goedhart & C.J.F. Ter Braak

Does regression by partial least squares with leave-out options

contents previous next

Options

`PRINT` = *string token*	Printed output required (`press`); default `press`
`NPLS` = *scalar*	The maximum number of PLS dimensions to be fitted; default 5 or the number of predictors if smaller than 5
`LEAVE` = *scalar* or *factor*	Determines leave-out groups. If a scalar is specified, groups of size `LEAVE` are formed in a systematic way. For `LEAVE=0` no units are left out; default 0, i.e. no units are left out
`STORE` = *scalar* or *variate*	Defines dimensions for which results are stored. If `STORE` is set to a variate, it must contain increasing values. For `STORE=0` results are stored for dimensions 1 ... `NPLS`. If `STORE=NPLS` then `VARBETA` is a variance-covariance matrix; default 0

Parameters

`X` = *pointers*	Pointer containing the predictor variables
`RESIDUALS` = *pointers*	Pointer to save variates with the residuals of the response variable for dimensions in `STORE`. Residuals are leave-out or resubstitution residuals depending on the way units are left out
`PRESS` = *variates*	To save the mean prediction sum of squares. `PRESS` is calculated over all units or, if `LEAVE` specifies two leave-out groups, over units in the second group
`BETA` = *pointers*	Pointer to save variates containing the regression coefficients or, if `LEAVE` specifies more than two leave-out groups, jackknife regression coefficients for dimensions in `STORE`
`VARBETA` = *pointers*	Pointer to save variates containing the jackknife variances of `BETA` or pointer to save a symmetric matrix containing the jackknife variance-covariance matrix of `BETA`, both for dimensions in `STORE`
`CONSTANT` = *pointers*	Pointer to save scalars containing the constant in the regression model for dimensions in `STORE`. `BETA` must be stored as well
`COEFFICIENTS` = *pointers*	Pointer to save variates containing the coefficients to calculate PLS-scores from the predictors, stored for dimensions 1 ... `NPLS`
`MAHALANOBIS` = *variates*	To save the Mahalanobis squared distance calculated with respect to `NPLS` scores. Distances are leave-out or resubstitution distances depending on the way units are left out

Description

Procedure `RPLS` does regression by partial least squares (PLS). PLS is useful in regression problems with many predictors or when predictors show high collinearity. These problems typically occur in observational studies and in multivariate calibration. PLS is an improvement over the earlier method of principal components regression (PCR). PCR is a two-stage method. At the first stage principal components of the predictor variables are formed to account for most of the variation in the predictors while at the second stage the response variable is regressed on these principal components. In PLS the two stages are combined so as to yield large predictive power with fewer dimensions than PCR. PLS sequentially forms orthogonal linear combinations of predictor variables with weights proportional to their covariance with the response variable when fitting the first dimension, and with the residual of the response variable when fitting subsequent dimensions. A description of the PLS algorithm and further references are given in Næs, Irgens & Martens (1986).

A call to `RPLS` must be preceded by a `MODEL` statement in which the response variable is specified. Only the first response variable is analysed and the `WEIGHTS`, `OFFSET` and `GROUPS` options of `MODEL`, if specified, are ignored for the PLS fit. Generalized models are not allowed. The number of PLS dimensions fitted is determined by option NPLS while the dimensions for which results are stored can be controlled by the STORE option. Option LEAVE defines the leave-out groups. If LEAVE is set to a scalar, say *m*, the first *m* units form the first group, the second *m* the second group, etc. LEAVE determines the way in which RESIDUALS, BETA, CONSTANT, MAHALANOBIS and PRESS are calculated. Three situations can be distinguished:

1. If LEAVE=0 PLS is performed for all units and BETA and CONSTANT are calculated accordingly. PRESS, RESIDUALS and MAHALANOBIS are calculated by resubstitution.
2. If LEAVE is set to a factor with two levels or set to a scalar which defines two groups, the first group is taken as training set and the second as evaluation set. BETA and CONSTANT are calculated for the training set. RESIDUALS and MAHALANOBIS are calculated by resubstitution for the training set and by leave-out for units in the evaluation set. PRESS is calculated over units in the evaluation set only.
3. If LEAVE is set to a factor with more than two levels or set to a scalar defining more than 2 groups, then all results are calculated by means of leave-out. Every group is subsequently left out from the analysis and leave-out RESIDUALS and MAHALANOBIS are calculated. PRESS is calculated over all leave-out residuals. BETA stores the jackknife regression coefficients while CONSTANT is set to missing values. In this case jackknife variances of BETA can be stored by means of VARBETA. A jackknife variance-covariance matrix of BETA can be stored by setting STORE equal to NPLS; the only dimension stored is then NPLS.

The COEFFICIENTS can be used to calculate PLS-scores from the predictor variables (see example 2). COEFFICIENTS are based on all units in situation 1 and 3, and in situation 2 based on the training set only. PRINT controls the printing of PRESS and of the square root of PRESS.

Method

The procedure passes the problem to a C# .NET Framework 4 executable by using the SUSPEND directive. The PLS algorithm can be formulated in different ways. Næs et al. (1986) gives two algorithms, with and without orthogonal scores. However, both algorithms need a time consuming updating step. Wold et al. (1984) showed that PLS essentially is a conjugate gradient algorithm for linear least squares problems. From the available algorithms, reviewed by Paige and Saunders (1982), the CGLS algorithm appeared to be a good compromise between numerical accuracy and speed (required for leave-out methods). The CGLS algorithm is implemented in C# using double precision arithmetic throughout. Numerical inaccuracy is most likely to affect the statistically irrelevant dimensions only.

Variables are standardized to zero mean and unit sum of squares before calling CGLS; the results are backtransformed to the original scale. RPLS thus implements PLS with "auto scaling" of the predictor variables (Wold et al., 1984). When units are left out, variables are re-standardized for the remaining units.

The calculation of scores from COEFFICIENTS in GenStat is numerically unstable, especially for higher dimensions. The computation can be checked by comparing the residuals of the regression of the response variable on the scores with the residuals produced by RPLS (see example 2).

The Fortran routine performs several checks. If one of these checks fails a fault code with an explanatory message is printed and the rest of the job will be ignored.

Action with RESTRICT

Only the response variate can be restricted. The analysis is restricted accordingly. The predictor variables and the leave-out factor must not be restricted. So if the leave-out factor only has two levels in the restricted set, the group with the first level is taken as the training set and the remaining units as the evaluation set. Parameters RESIDUALS and MAHALANOBIS are not restricted on output.

References

Næs, T., Irgens, C. and Martens, H. (1986). Comparison of Linear Statistical Methods for Calibration of NIR Instruments. *Applied Statistics*, **35**, 195-206.

Wold, S., Ruhe, A., Wold, W. and Dunn, W.J. (1984). The Collinearity Problem in Linear Regression. The Partial Least Squares (PLS) Approach to Generalized Inverses. *SIAM Journal of Statistical Computing*, **5**, 735-743.

Paige, C.C. and Saunders, M.A. (1982). A Bidiagonalization Algorithm for Sparse Linear Equations and Least Squares Problems. *ACM Trans. Math. Software,* **8**, 43-71.

Næs, T. (1985). Multivariate Calibration when the Error Covariance Matrix is Structured. *Technometrics*, **27**, 301-311.

Procedures Used

The `BIOMETRIS` procedure is used to retrieve the filename of the external C# executable.

Similar Procedures

`PLS` fits a partial least squares regression model.

Example

```
CAPTION   'RPLS example 1', !t('Analysis of data from Naes (1985).', \
          'Perform leave-one-out on the samples in training and evaluation', \
          'set. Store the jackknife regression coefficients and variances', \
          'for PLS-dimensions 3-6 and calculate t-values. Also calculate', \
          'and print the jackknife variance-covariance matrix of the', \
          'jackknife regression coefficients for dimension 4.'), ' ' ; \
          STYLE=meta, 2(plain)
UNIT      [45]
FACTOR    [LAB=!T(training,evaluation,outlier) ; VAL=20(1),18(2),7(3)] set
READ      fat, nir[1...9]
  30.4  1.1398 1.0568  .9092  .8779  .7183  .5810  .6195  .6472  .3779
  41.8  1.4455 1.3258 1.1355 1.1067  .9213  .7834  .8363  .8661  .5015
  44.1  1.4606 1.3413 1.1519 1.1216  .9333  .7975  .8502  .8807  .5099
  42.7  1.5637 1.4363 1.2334 1.2019  .9941  .8534  .9116  .9440  .5370
  38.7  1.3597 1.2561 1.0818 1.0487  .8614  .7275  .7761  .8060  .4643
  39.9  1.4400 1.3299 1.1479 1.1154  .9248  .7791  .8303  .8614  .5036
  35.9  1.4534 1.3433 1.1603 1.1257  .9225  .7677  .8197  .8529  .4878
  40.8  1.5675 1.4400 1.2434 1.2107 1.0182  .8651  .9202  .9517  .5644
  38.6  1.3571 1.2475 1.0723 1.0409  .8595  .7180  .7663  .7964  .4619
  41.6  1.4391 1.3160 1.1262 1.0979  .9297  .7696  .8215  .8507  .5016
  44.8  1.5790 1.4431 1.2339 1.2049 1.0026  .8691  .9280  .9588  .5504
  44.8  1.7891 1.6593 1.4526 1.4197 1.1967 1.0389 1.1035 1.1379  .6604
  43.6  1.6179 1.4921 1.2910 1.2602 1.0570  .9143  .9739 1.0056  .5815
  43.1  1.5615 1.4343 1.2343 1.2027 1.0000  .8657  .9232  .9542  .5503
  39.6  1.4028 1.2825 1.0944 1.0621  .8749  .7349  .7864  .8173  .4687
  45.2  1.5438 1.4282 1.2416 1.2150 1.0290  .8966  .9520  .9811  .5822
  41.8  1.5455 1.4256 1.2309 1.2018  .9986  .8554  .9121  .9432  .5461
  43.3  1.6107 1.4851 1.2871 1.2569 1.0545  .9113  .9678  .9992  .5888
  41.6  1.4498 1.3410 1.1650 1.1361  .9572  .8129  .8660  .8961  .5194
  31.6  1.2834 1.1894 1.0246  .9894  .8034  .6483  .6926  .7240  .4122
  43.0  1.4015 1.2830 1.0962 1.0678  .8795  .7597  .8109  .8394  .4808
  35.9  1.3636 1.2630 1.0925 1.0618  .8868  .7313  .7790  .8089  .4773
  36.0  1.3921 1.2863 1.1081 1.0751  .8876  .7384  .7868  .8173  .4782
  42.3  1.4416 1.3211 1.1259 1.0938  .8945  .7651  .8186  .8492  .4808
  43.3  1.4938 1.3744 1.1893 1.1612  .9882  .8375  .8912  .9206  .5466
  45.4  1.4985 1.3670 1.1671 1.1399  .9500  .8214  .8765  .9059  .5264
  40.7  1.6116 1.4886 1.2867 1.2518 1.0367  .8858  .9440  .9780  .5602
  40.4  1.4787 1.3565 1.1679 1.1379  .9583  .8047  .8570  .8870  .5276
  44.5  1.6614 1.5272 1.3244 1.2933 1.0961  .9508 1.0107 1.0412  .6116
  46.3  1.5601 1.4348 1.2445 1.2186 1.0440  .9056  .9620  .9895  .5921
  39.1  1.5353 1.4164 1.2241 1.1923  .9970  .8429  .8983  .9302  .5418
  37.6  1.3876 1.2804 1.1012 1.0679  .8758  .7356  .7851  .8157  .4706
```

```
  37.1  1.2840 1.1825 1.0131  .9828  .8000  .6708  .7160  .7450  .4277
  39.4  1.4004 1.2862 1.1019 1.0696  .8826  .7444  .7951  .8252  .4733
  41.6  1.3602 1.2559 1.0892 1.0601  .8857  .7578  .8068  .8354  .4899
  36.4  1.3841 1.2807 1.1059 1.0724  .8841  .7383  .7868  .8175  .4766
  40.7  1.5202 1.3938 1.1908 1.1568  .9555  .8088  .8641  .8956  .5139
  36.3  1.3512 1.2504 1.0773 1.0444  .8573  .7090  .7580  .7891  .4472
  48.7  1.9052 1.7616 1.5381 1.5096 1.2798 1.1353 1.2070 1.2405  .7061
  48.2  1.9421 1.7861 1.5544 1.5202 1.2821 1.1278 1.2012 1.2347  .7020
  48.9  1.8434 1.6965 1.4717 1.4449 1.2229 1.0968 1.1653 1.1956  .6932
  43.3  1.5232 1.4021 1.2134 1.1880 1.0274  .8628  .9172  .9459  .5755
  46.8  2.2651 2.0703 1.7860 1.7502 1.4664 1.2948 1.3804 1.4192  .7852
  40.7  2.2558 2.0819 1.8088 1.7601 1.4652 1.2189 1.3010 1.3441  .7453
  42.7  2.0981 1.9512 1.7160 1.6776 1.4222 1.2142 1.2920 1.3301  .7602  :
RESTRICT  fat ; set.NE.3
MODEL     fat
RPLS      [NPLS=6 ; LEAVE=1 ; STORE=!(3...6)] X=nir ; BETA=jackbeta ; \
          VARBETA=jackvar
CALCULATE tvalue[3...6] = jackbeta[]/SQRT(jackvar[])
PRINT     jackbeta[]
PRINT     tvalue[]
CAPTION   'RPLS example 2', !t('Calibrate on samples 1-20; evaluate on', \
          'samples 21-38. Calculate PLS-scores and residuals (resB) from',\
          'the regression of fat on the scores. Compare resB with', \
          'the residuals (resA) obtained by RPLS. Note that the difference',\
          'between resA and resB for dimension 6 is caused by collinearity', \
          'among the predictors.'), ' ' ; STYLE=meta, 2(plain)
RPLS      [NPLS=6 ; LEAVE=set] X=nir ; RESIDUALS=resA ; BETA=beta ; \
          COEFFICIENTS=coef
RESTRICT  fat, nir[], set, resA[]
PRINT     beta[]
PRINT     coef[]
RESTRICT  fat ; set.EQ.1
MODEL     fat
FOR  ii=1...6 ; iiplus=2...7
  CALCULATE dummy[1...9] = (s[1...9]=coef[ii]$[1...9])*nir[1...9]
  CALCULATE score[ii] = VSUM(dummy)
  FIT       [PRINT=*] score[1...ii]
  RKEEP     ESTIMATES=esti
  CALCULATE dummy[1...ii] = esti$[2...iiplus]*score[1...ii]
  CALCULATE fitB[ii] = esti$[1] + VSUM( !P(dummy[1...ii]) )
ENDFOR
RESTRICT  fat
CALCULATE resB[1...6] = fat - fitB[]
PRINT     set, resA[4...6], resB[4...6] ; FIELD=14,6(9) ; DECI=2
```

RSELECT

P.W. Goedhart

Selects best subsets of predictor variables in regression

contents previous next

Options

`CRITERION` = *string token*	Criterion for selecting best subsets (`r2`, `adjusted`, `cp`); default `r2`
`CONSTANT` = *string token*	How to treat the constant (`estimate`, `omit`); default `estimate`
`PENALTY` = *scalar*	Penalty in interval (0,∞) for Mallows C_p; default 2
`NBESTMODELS` = *scalar*	Number of subsets desired for each size of subset when `CRITERION=r2` (default 5), in total when `CRITERION` is set to `adjusted` or `cp` (default 30). The total number of subsets to be selected should not exceed 150
`TOLERANCE` = *scalar*	Minimum tolerance in order to exclude subsets that are collinear. The default value is k²*2**(-28) where k is the total number of `FREE` and `COVARIATES` predictors. A value less than the default is replaced by the default. The tolerance should be in the interval [0,1)
`NPRINT` = *scalar*	The maximum size of subsets for which output is printed; default 30
`MEANSQUARE` = *scalar*	Mean square of residuals for calculation of C_p. A value smaller than or equal to 0 is replaced by the mean square residual of the full model; default 0
`FORCED` = *formula*	Model formula that is fitted to the response variate before best subsets are selected
`COVARIATES` = *variates*	Predictor variables that should be included in each model
`RESULTS` = *pointer*	Pointer to save 5 variates containing the number of included predictors, the three criteria and the minimum tolerance for the selected subsets
`TVALUES` = *pointer*	Pointer to save variates containing the t-values of regression coefficients for the selected subsets. A missing value indicates that the corresponding predictor is not in the selected subset
`SUBSETS` = *pointer*	Pointer to save pointers with the selected subsets of predictor variables `FREE` and `COVARIATES`

Parameters

`FREE` = *variates*	Predictor variables from which best subsets are selected

Description

There are various methods for choosing a regression model when there are many predictor variables, see e.g. Montgomery and Peck (1992). GenStat directive `STEP`, used in a `FOR` loop, provides forward selection, backward elimination and stepwise regression. However these methods result in only one model and alternative models, with an equivalent or even better fit, are easily overlooked. Moreover, in most applications the particular predictors that effect the response and the directions of their effects are of intrinsic interest and then selection of just one well-fitting model is unsatisfactory and possibly misleading. To overcome this, `RSELECT` evaluates all possible subsets of predictor variables and selects a small number of best subsets. `RSELECT` should be used with caution, especially when the number of predictors is large in comparison with the number of units. In this case uncritical use of `RSELECT` might lead to models which appear to have a lot of explanatory power, but contain noise variables only, see e.g. Flack and Chang (1987). Predictors should therefore not be selected on the basis of a statistical analysis alone.

Identification of the best subsets depends upon the criterion used for measuring goodness of fit. The three criteria employed in `RSELECT` are widely used and are defined as follows:

R^2	100 * (1 – RSS / SSY)
Adjusted R^2	100 * (1 – (n – 1) * RSS / (SSY * (n – p)))
Mallows C_p	RSS / RMSFULL + 2*p – n

in which

RSS	is the residual sum of squares for the subset at hand;
SSY	is the sum of squares about the mean of the response variate;
p	is the number of fitted parameters (including the intercept);
n	is the number of units;
RMSFULL	is the residual mean square for the full model.

When R^2 is used as selection criterion, there is no penalty for adding a predictor. R^2 always improves with the addition of a predictor. When adjusted R^2 or C_p is employed, there is a penalty for adding a variable. Adjusted R^2 improves when the F-ratio due to the addition of the predictor is larger than 1, while C_p improves when the F-ratio is larger than 2. Clearly, C_p is the more conservative criterion and will tend to select smaller subsets as compared to R^2 and adjusted R^2. The definition of C_p can be altered by setting options `PENALTY` and `MEANSQUARE`, in which case

C_p = RSS / `MEANSQUARE` + `PENALTY*`p - n.

In this case C_p improves when the F-ratio is larger than `PENALTY`.

An advantage of using R^2 as the selection criterion is that the best subsets within **each** size of subset are selected. In the case of adjusted R^2 or C_p best subsets are selected regardless of the subset size. The `CRITERION` option controls which of the three criteria is used. The number of selected subsets is defined by the `NBESTMODELS` option, while the `NPRINT` option controls the maximum size of subset for which output is produced. Default is to print all selected subsets. The printed output of `RSELECT` is adjusted to the width of the output file. Output might not be transparent if the output width is too small.

Best subsets are selected subject to an optional constraint, set by the `TOLERANCE` option, on the degree of correlation permitted among the predictor variables of a subset. For this purpose the minimum tolerance of a subset is used. This is defined as 1 minus the maximum of all the multiple correlations between individual predictors in the subset and the remaining predictors in that subset. The minimum tolerance is thus a measure of the degree of multicollinearity in the subset with small values indicating collinearity. If the minimum tolerance for a subset is smaller than the setting of `TOLERANCE`, that subset is omitted and a message is printed. Subsets with small tolerances are often unstable and may perform poorly when they are used for prediction purposes. Therefore, if `RSELECT` only selects subsets with small tolerances, a deeper search can be forced by specifying a larger value for `TOLERANCE`.

A call to `RSELECT` must be preceded by a `MODEL` statement which defines the response variate and, if required, a vector of weights and an offset. The `MODEL` directive should not specify the `GROUPS` option. Only the first response variate is analysed and generalized linear regression models are not allowed. The `FREE` parameter specifies the list of predictors from which best subsets are selected according to the chosen criterion. For each selected subset the three criteria are printed along with the minimum tolerance and t-values of regression coefficients of included predictors. Negative values for the criteria are not printed and C_p is truncated at 999.99. In order to keep output concise, t-values are also truncated.

It is sometimes desirable to include some predictors in each model and to investigate whether other predictors should be added to the model. Predictors that must be included in each model can be specified by means of the `FORCED` or the `COVARIATES` option. The `COVARIATES` option should be set to a list of variates and t-values are printed for these predictors. Alternatively, the `FORCED` option can be set to any formula, `FORCED` may thus contain factors and interactions as well as variates. The `FORCED` formula is fitted first to the response variate and to the `FREE` and `COVARIATES` predictors and the procedure continues with the residuals from these regressions. Consequently, the minimum tolerance is calculated for predictor variables allowing for the `FORCED` formula. This implies that the minimum tolerance depends on whether a `FORCED` formula is used or not. Moreover, t-values for the parameters associated with the `FORCED` formula are not printed. Note that if `NPRINT` is smaller than or equal to the number of `COVARIATES` variates, no output is produced.

The `FREE` and `COVARIATES` list of variates and the `FORCED` formula should be mutually exclusive. The total number of `FREE` and `COVARIATES` predictors should not exceed 30. The `CONSTANT` option controls whether the constant parameter is included in the model.

Cases with one or more missing values in the response variate, weight vector or any term in the full model are excluded from the analysis. This implies that, when terms have missing values for different units, `FIT` used on a subset of terms may give different results than `RSELECT`.

Options `RESULTS`, `TVALUES` and `SUBSETS` allow saving of the output. The `RESULTS` option saves a pointer with 5 variates, containing the number of included predictors, the three criteria and the minimum tolerance for the selected subsets. The t-values of regression coefficients and the formulae of the selected subsets can be saved by means of parameters `TVALUES` and `SUBSETS` respectively. All selected models are saved regardless of the setting of `NPRINT`. The saved criteria and t-values are not truncated.

Method

If a `FORCED` formula is specified, the response variate and `FREE` and `COVARIATES` predictors are regressed on the `FORCED` formula and the response variate and predictors are replaced by the residuals of these regressions. The full model, including all `COVARIATES` and `FREE` predictors, is then fitted in order to exclude aliased predictors from the analysis. The double precision regression work structure is saved using the `FSSPM` directive and passed to an external C# .NET Framework 4 executable by using the `SUSPEND` directive. The C# program calls subroutine Screen (Ter Braak and Groeneveld, 1982) which was written in Fortran but ported to C#.

Screen is the 1981 double precision version of a branch and bound algorithm for subset selection developed by Furnival and Wilson. They claim that this version is twice as fast as the 1974 version (Furnival and Wilson, 1974), requires much less storage and handles problems with rank deficient data in a more satisfactory manner. The algorithm stores the largest discrepancy observed in the value of the selection criterion for a number of numerical consistency checks. `RSELECT` prints this discrepancy as an indication of the numerical accuracy of the algorithm. The criteria, minimum tolerance and t-values are returned to GenStat. If necessary, the criteria and t-values are adjusted to incorporate effects of the fitting of a `FORCED` formula, the minimum tolerance is not adjusted. The procedure itself deals mainly with checking of options and parameters and with input and output of the Fortran program.

Action with RESTRICT

Only the response variate can be restricted. The analysis is restricted accordingly. The vector of weights, the offset and terms in `FREE`, `FORCED` and `COVARIATES` must not be restricted.

References

Flack, V.F. and Chang, P.C. (1987). Frequency of selecting noise variables in subset regression analysis: a simulation study. *The American Statistician*, **41**, 84-86.

Furnival, G.M. and Wilson, R.W. (1974). Regression by leaps and bounds. *Technometrics*, **16**, 499-511.

Montgomery, D.C. and Peck, E.A. (1992). *Introduction to linear regression analysis, second edition*. Wiley. New York.

Ter Braak, C.J.F. and Groeneveld, A. (1982). *SUBSEL - een Fortran programma voor "SUBset SELection" in regressiemodellen gebaseerd op subroutines van Furnival en Wilson*. IWIS rapport B 82 ST 79 41. Wageningen. The Netherlands.

Procedures Used

The `BIOMETRIS` procedure is used to retrieve the filename of the external C# executable.The `VEQUATE` procedure is used when the `SUBSETS` option is set.

Similar Procedures

`RSCREEN` performs screening tests for generalized or multivariate linear models with many predictors. `RSEARCH` helps search through models for a regression or generalized linear model. `VSEARCH` helps search through models for a generalized linear mixed model (GLMM).

Example

```
CAPTION   'RSELECT example 1', !t('Data taken from Montgomery and Peck', \
          '(1992), page 277.'), ' ' ; STYLE=meta, 2(plain)
UNIT      [13]
READ      x1, x2, x3, x4, response
     7  26   6  60   78.5     1  29  15  52   74.3    11  56   8  20  104.3
    11  31   8  47   87.6     7  52   6  33   95.9    11  55   9  22  109.2
     3  71  17   6  102.7     1  31  22  44   72.5     2  54  18  22   93.1
    21  47   4  26  115.9     1  40  23  34   83.8    11  66   9  12  113.3
    10  68   8  12  109.4  :
MODEL     response
RSELECT   x1,x2,x3,x4
CAPTION   'RSELECT example 2', !t('Shows the difference in using the FORCED',\
          'or COVARIATES option for predictors that should be included in',\
          'each model'), ' ' ; STYLE=meta, 2(plain)
RSELECT   [FORCED=x4]    FREE=x1,x2,x3
RSELECT   [COVARIATES=x4] FREE=x1,x2,x3
CAPTION   'RSELECT example 3', !t('Shows how the RESULTS and SUBSETS', \
          'options can be used to FIT all selected models', \
          'with 2 predictors'), ' ' ; STYLE=meta, 2(plain)
RSELECT   [NPRINT=* ; RESULTS=results ; SUBSETS=subsets] x1,x2,x3,x4
RESTRICT  results[1] ; CONDITION=results[1].EQ.2 ; SAVESET=saveset
FOR       ii=#saveset
  FIT     subsets[ii][]
ENDFOR
CAPTION   'RSELECT example 4', !t('Shows that when the number of predictors',\
          'is large compared with the number of units, models with noise',\
          'variables only may appear to have a lot of explanatory power.'), \
          ' ' ; STYLE=meta, 2(plain)
VARIATE   [NVALUES=12] y, noise[1...10]
CALCULATE y, noise[] = URAND(912439,10(0) ; 12)
MODEL     y
RSELECT   noise[]
```

RUNCERTAINTY

M.J.W. Jansen, J.C.M. Withagen & J.T.N.M. Thissen

Calculates contributions of model inputs to the variance of a model output

contents previous next

Options

PRINT = *string tokens*	What to print (fullmodel, uncertainty); default fullmodel, uncertainty.
PLOT = *string token*	Graphical output required (histogram); default *
CURVE = *string token*	Type of curve to be fitted (linear, spline); default linear
ESTIMATES = *variate*	To save regression coefficients of all X variates (for CURVE=linear)
BOTTOM% = *variate*	To save bottom marginal variances as percentage of the variance of the model output. Increase of percentage variance accounted for when an X structure is last to be added
TOP% = *variate*	To save top marginal variances as percentage of variance of the variance of the model output. Percentage variance accounted for when an X structure is the only one to be fitted.
ADJUSTEDR2 = *scalar*	To save adjusted percentage of variance accounted for by all X variates

Parameters

X = *pointers* or *variates*	Set of model inputs for which uncertainty contributions are to be calculated. If a pointer is specified it must only point to variates
DF = *scalars*	Effective degrees of freedom of the smoothing splines to fit for each X structure; default 2
FITTEDVALUES = *variates*	Variates to store the fitted values for each X structure when that input is the only one to be fitted

Description

Procedure RUNCERTAINTY performs uncertainty analysis given (1) a sample of model inputs from a joint distribution representing the uncertainty about these inputs and (2) a corresponding sample of the model output studied. The model output, given its inputs, may have been produced by specialised modelling software. The procedure calculates the contributions to the variance of the model output from individual or pooled model inputs by means of regression. These contributions are expressed as percentages of the variance of the model output. The top marginal variance of a model input is calculated as the percentage of variance accounted for when that input is the only one to be fitted; it is an approximation of the correlation ratio. The bottom marginal variance of an input is calculated as the increase of variance accounted for when that input is the last to be added to all other inputs. The calculation is successful if the percentage of variance accounted for by all inputs is close to 100, since the analysis only accounts for that part of the variance of the output that is explained by the regression (thus interactions between inputs are not considered). See Jansen et al (2002) and Saltelli et al (2000) for a detailed account of uncertainty analysis.

A call to RUNCERTAINTY must be preceded by a MODEL statement which defines the response variate with the model outputs. Only the first response variate is analysed and options other than WEIGHTS should not be set in the MODEL statement. Generalized linear models are not allowed. The model inputs are specified by the X parameter that can consist of variates or pointers to one or more variates. If a pointer is specified the total contribution of the variates of the pointer is calculated. The calculation applies multiple linear regression or spline regression of Y on the X structures plus a constant term. The choice between linear and spline regression can be made by means of the CURVE option. When using CURVE=spline, the degrees of freedom of the smoothing spline can be set separately for each X structure by means of the DF parameter. On output the full model has been fitted, and RKEEP and RDISPLAY can be used to further store and display the fit of the full model.

Cases with one or more missing values in the response variate, weight vector or any term in the full model are excluded from the analysis. This implies that, when terms have missing values for different units, FIT used on a subset of model inputs may give different results than RUNCERTAINTY.

The option setting PRINT=fullmodel prints the fit of the full model while suppressing all warning messages. Setting PRINT=uncertainty prints the top and bottom marginal %variances of the X structures and, in case CURVE=linear, the parameter estimates of the full model. The option setting PLOT=histogram option draws a histogram of the top and bottom marginal %variances side by side for each of the X structures. The results of the uncertainty analysis can be saved by means of options ESTIMATES (in case CURVE=linear), BOTTOM%, TOP% and ADJUSTEDR2. The fitted values of the models with individual X structures only (pointers and/or variates) can be saved by means of the FITTEDVALUES parameter. These fittedvalues correspond to the top marginal %variances.

Method

The procedure calculates the percentage of variance accounted for the relevant regressions. The top marginal %variance for an input X is calculated as 100(vary-rmstop)/vary, where vary is the variance of the response and rmstop is the residual mean square of the model with only input X. The bottom marginal %variance for an input X equals 100(rmsbottom-rmsall)/vary, where rmsall is the residual mean square of the full model with all inputs, and rmsbottom is the residual mean square of the full model without input X. A TERMS statement in the procedure deals with missing values in the X variates.

Action with RESTRICT

Only the response variate can be restricted. The analysis is restricted accordingly. Restrictions on the X structures are not allowed. The saved FITTEDVALUES variates will be unrestricted, but only units not excluded by the restriction will have values.

References

Jansen M.J.W. ,Withagen J.C.M. & Thissen J.T.N.M. (2002). *USAGE: uncertainty and sensitivity analysis in a GenStat environment. Manual. Version 2.1*. Wageningen: Biometris (in prep).

Saltelli, A. & Chan, K. & Scott, E.M. (2000; eds.). *Sensitivity analysis*. Chichester: Wiley.

Procedures Used

FEXPAND.

Similar procedures

GMULTIVARIATE and GUNITCUBE can be used to generate random inputs. RSELECT selects best subsets of predictor variables in regression. RSCREEN performs screening tests for generalised or multivariate linear models. RSEARCH helps search through models for a regression or generalised linear model.

Example

```
CAPTION      'RUNCERTAINTY example' ; STYLE=meta
POINTER      par  ; !p(a0, a1, a2)
POINTER      soil ; !p(ph, cd)
READ         par[1...3], esp, soil[1,2], lcdp ; DECIMALS=1
 59 42 43 69 59 66 2199    55 39 48 52 57 54 1726    60 59 50 46 58 43 1631
 53 43 49 53 50 30 1134    49 48 71 52 29 73 1292    64 52 44 55 51 43 1411
 67 67 32 64 62 53 2042    51 49 52 51 47 44 1224    47 33 54 48 30 51 1043
 44 45 48 52 34 42  870    43 44 54 59 64 66 2028    55 40 38 59 46 62 1435
 50 50 48 42 56 47 1374    64 69 54 55 61 50 2004    47 55 57 46 59 40 1405
 39 62 58 53 68 50 1894    61 39 59 47 35 47  948    73 37 52 41 45 38  992
 40 61 50 64 58 49 1616    44 55 56 51 52 50 1388    48 50 41 35 42 60 1167
 51 48 44 58 45 54 1147    76 49 48 48 50 37 1190    47 51 46 28 66 64 1973
 53 44 47 65 44 64 1354  :
MODEL        lcdp
RUNCERTAINTY [CURVE=linear] x=par,esp,soil
RUNCERTAINTY [CURVE=spline] x=par,esp,soil ; DF=1,1,2
```

SETDEVICE

M.P. Boer, J.T.N.M. Thissen & P.W. Goedhart

Opens a graphical file on the basis of a file extension

contents previous next

Options

PRINT = *string token*	What to print (filename); default *
SURNAME = *text*	Surname of graphical file to open; default * uses the surname of the file on the current input channel
EXTENSION = *string token*	Extension of graphical file to create (hpg, eps, emf, jpg, tif, png, gmf or bmp); default png
SUBDIRECTORY = *text*	Subdirectory below the directory of the file on the current input channel in which the graphical files will be opened; default * does not use a subdirectory
FIRSTNUMBER = *scalar*	First sequence number for the graphical file name; default 1
NDIGITS = *scalar*	Number of zeros with which the sequence number of the graphics file is padded; default 2
PALETTE = *string token*	How to represent colour (monotone, greyscale, grayscale, colour); default colour
RESOLUTION = *scalar*	Specifies the height of the saved image in pixels; default 800
SEQUENCERESET = *string token*	Whether to reset the sequence of graphical files (yes, no) default no
CREATEFILE = *scalar*	Logical expression indicating whether (1) or not (0) to create a new graphical file; default 1 creates a new graphics file
SAVEFILENAME = *text*	Saves the filename of the graphical file. This is only set to a new filename when the logical expression CREATEFILE is set to true

Parameters

FILENAME = *texts*	Name of the graphical file including one of the possible extensions .hpg, .eps, .emf, .jpg, .tif, .png, .gmf or .bmp
NUMBER = *scalars*	Saves the device number corresponding to the graphical format specified by parameter FILENAME or, if this is not set, by option setting EXTENSION

Description

Procedure SETDEVICE can be used in two different ways depending on whether the FILENAME parameter is set. In case FILENAME is set all options except PALETTE and RESOLUTION are ignored and a graphical file is opened with that name. The device number is set on the basis of the extension of the given filename. The table below shows the correspondence between the file extension and the device number.

Extension	Device	Type of file
.hpg	4	HPGL file
.eps	5	Encapsulated PostScript file
.emf	6	Windows Enhanced Meta File
.jpg	7	JPEG file
.tif	8	TIFF file
.png	9	Portable Network Graphics file
.gmf	10	GenStat Meta File
.bmp	11	Windows BitMaP file

The default setting PALETTE=colour produces coloured output and enables the use of the COLOUR directive to specify exactly the composition of the colours. Alternatively, the monotone setting forces all colours to be mapped to colour 1. 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 saved image in pixels. This is equivalent to setting the image resolution in the Options menu of the GenStat Graphics Viewer.

When the FILENAME parameter is not specified, most option settings determine automatic opening and closing of a sequence of graphics files. In case the SURNAME option is not set, the directory and surname of the file on the current input channel will be used to name the graphics file which will be opened. In case the SURNAME option is set, the given surname and the working directory will be used. The directory can be appended by a subdirectory, one level deep only, as specified by the SUBDIRECTORY option. The filename is appended with '–', a sequence number and the EXTENSION option setting to give a new filename. The first sequence number is given by the setting of the FIRSTNUMBER option, and successive calls to SETDEVICE will update the sequence number. The NDIGITS option can be used to pad the sequence number with extra zeros. The corresponding graphics channel, according to the EXTENSION setting in the table above, is then closed and a new graphics file with the new filename is opened. The new graphics file will use the current settings of the PALETTE and RESOLUTION option. The SEQUENCERESET option can be used to reset the sequence of filenames. Note that the option settings SURNAME, EXTENSION, SUBDIRECTORY, FIRSTNUMBER and NDIGITS are only relevant the first time SETDEVICE is called or when SEQUENCERESET=yes. In other cases the settings of a previous call to SETDEVICE are used. However the options settings of PALETTE and RESOLUTION are relevant for every call to SETDEVICE.

By default a new plot file is always created, but this can be supressed by setting the CREATEFILE option to zero. This is especially convenient when multiple windows are used in a single graph, see for instance the example program. The filename can be saved by means of the SAVEFILENAME option; this will only be set to a new filename when the logical expression CREATEFILE is set to 1 (true). Finally note that the last graphical file in a program will only be created when a STOP directive is encountered, or when the corresponding graphical device is explicitly. Also note that re-running the same program without a STOP directive will create duplicate graphical files. It is therefor adviced to insert a STOP directive when using SETDEVICE as in the example program.

Method

A GenStat WORKSPACE with name 'BiometrisSetDevice' is used to save the current option settings between successive calls to the procedure.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

SFILENAME.

Similar Procedures

None.

Example

```
CAPTION   'SETDEVICE example with automatic creation of graphics files', \
          !t('Setting the scalar createfile to 0 will display', \
          'the plots in the GenStat Graphics Viewer.', \
          'The plots are examples of a spatial Poisson process.', '') ; \
          STYLE=meta,plain
SCALAR    createfile ; 1
FFRAME    [ROWS=4 ; COLUMNS=4 ; MARGIN=small ; SQUARESHAPES=yes] \
          NGRAPHS=ngraphs; SSCREEN=screen
XAXIS     1...#ngraphs ; LOWER=0 ; UPPER=1 ; MARKS=!(10)
YAXIS     1...#ngraphs ; LOWER=0 ; UPPER=1 ; MARKS=!(10)
PEN       NUMBER=1 ; METHOD=point ; SYMBOLS=2 ; SIZE=0.3
CALCULATE initialize = URAND(8412825 ; 1)
SCALAR    win ; 0
CALCULATE ntimes = 2*ngraphs
FOR [NTIMES=ntimes ; INDEX=ii]
  CALCULATE yrandom, xrandom = URAND(0,0 ; 50)
  CALCULATE win = win + 1 - ngraphs*(win.eq.ngraphs)
  SETDEVICE [PRINT=filename ; CREATEFILE=createfile*(win.eq.1)] NUMBER=Device
  DGRAPH    [WINDOW=win ; KEYWINDOW=0 ; SCREEN=#screen[win]] yrandom ; xrandom
ENDFOR
CLOSE     CHANNEL=Device ; FILETYPE=graphics
XAXIS     [RESET=yes] 1...#ngraphs
YAXIS     [RESET=yes] 1...#ngraphs
FRAME     [RESET=yes] 1...#ngraphs
CAPTION   'SETDEVICE example using the FILENAME parameter' ; STYLE=meta
SETDEVICE 'SetFilename.png' ; NUMBER=Device
DGRAPH    [KEY=0] !(1...10)/11 ; !(1...10)/11
STOP
```

SFILENAME

P.W. Goedhart

Splits a filename, which is opened by GenStat or not, into substrings

contents previous next

Options

INPUTNAME = *text*	Filename to split into substrings; default is to use the CHANNEL and FILETYPE parameters
CASE = *string token*	Case to use for output structures (given, lower, upper, changed); default given leaves the case of each letter as given in the original string

Parameters

CHANNEL = *scalars*	Channel numbers for which the filenames must be retrieved; default 1
FILETYPE = *string tokens*	Type of each file (input, output, unformatted, backingstore, procedurelibrary, graphics); default input
OPEN = *scalars*	To indicate whether or not the corresponding channels are currently open (0=closed, 1=open)
NAME = *texts*	Saves the full name of the file, including the directory
DIRECTORY = *texts*	Saves the directory of the file, including the trailing slash
FILENAME = *texts*	Saves the name of the file, excluding the directory
SURNAME = *texts*	Saves the surname of the file, i.e. the name excluding the path, the period and the extension
EXTENSION = *texts*	Saves the extension of the file, excluding the leading period

Description

Procedure SFILENAME can be used in two different ways. The first use is to ascertain whether a particular channel is already in use and, if so, the name of the attached file is split into substrings. The channel for which substrings of filenames are required must be specified by means of the parameters CHANNEL and FILETYPE; the other parameters save the required information in data structures of the appropriate type. Text structures are only saved when the channel is open.

The second use is to split the setting of the INPUTNAME option; in that case the NAME parameter equals INPUTNAME and the OPEN parameter is set to missing.

The CASE option enables you to change the case of the letters of the output texts. By default, CASE=given leaves 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.

Method

The ENQUIRE directive is used to retrieve the OPEN and NAME parameters. The substrings are formed by text manipulation using CONCATENATE.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

None.

Similar Procedures

`QFILENAME` returns a single filename selected by means of a file open box on screen.

Example

```
CAPTION   'SFILENAME example' ; STYLE=meta
FOR filetype='input', 'output', 'backingstore'
  SFILENAME CHANNEL=1 ; FILETYPE=#filetype ; OPEN=open ; NAME=name ; \
            DIRECTORY=directory ; SURNAME=surname ; EXTENSION=extension
  IF open
      PRINT   [ORIENTATION=across ; RLWIDTH=12] name, directory, surname, \
              extension ; JUSTIFICATION=left ; SKIP=4
    ELSE
      PRINT   [IPRINT=*] '***** No', filetype, 'file open on CHANNEL=1'
  ENDIF
ENDFOR
TEXT      filename ; 'D:\\Windows\\System32\\*.dll'
SFILENAME [INPUTNAME=filename] DIRECTORY=directory ; SURNAME=surname ; \
          EXTENSION=extension
PRINT     [ORIENTATION=across ; RLWIDTH=12] filename, directory, surname, \
          extension ; JUSTIFICATION=left ; SKIP=4
```

SPECIALFUNCTION

P.W. Goedhart

Calculates a number of special mathematical functions

contents previous next

Options

None.

Parameters

X = *numerical structure*	Argument of special function
J0BESSEL = *numerical structure*	Saves the Bessel function of order 0 for arguments in X
J1BESSEL = *numerical structure*	Saves the Bessel function of order 1 for arguments in X
I0BESSEL = *numerical structure*	Saves the modified Bessel function of order 0 for arguments in X
I1BESSEL = *numerical structure*	Saves the modified Bessel function of order 1 for arguments in X
A1 = *numerical structure*	Saves the function $A_1(x)$ for arguments in X
A1INVERSE = *numeric. structure*	Saves the inverse of the function $A_1(x)$ for arguments in X
A1DERIVATIVE = *numerical structure*	Saves the first order derivative of the function $A_1(x)$ for arguments in X

Description

Procedure SPECIALFUNCTION can be used to calculate the following special mathematical functions:

- the Bessel functions of order 0 and 1, notated by J0(x) and J1(x) respectively;
- the modified Bessel functions of order 0 and 1, notated by $I_0(x)$ and $I_1(x)$ respectively;
- the function $A_1(x) = I_1(x) / I_0(x)$, its inverse and its first order derivative.

The argument for the Bessel functions and for $A_1(x)$ and its derivative must not be negative, while the argument for the inverse of $A_1(x)$ must be in the interval [0,1). For more information about the Bessel functions, see e.g. Abramowitz and Stegun (1964). The function $A_1(x)$ is especially useful in the analysis of circular data, see Fisher (1993).

The parameter X of the SPECIALFUNCTION procedure defines the arguments of the functions, using the notation given above. The function values can be saved by means of the other parameters; these are redefined to match the size and type of X.

All parameters must be either scalars, variates, matrices, symmetric matirces, diagonal matrices or tables. Pparameters must also have the same type.

Method

The (modified) Bessel functions employ algorithms described in Press *et al* (1992). The inverse of $A_1(x)$ is calculated by means of linear interpolation by calculating $A_1(x)$ for a suitable range of values x and using the fact that $\mathrm{Logit}(A_1(x))$ is almost linear in Log(x).

Action with RESTRICT

The arguments X can be restricted. The function values are restricted accordingly with missing values for units excluded by the restriction.

References

Abramowitz, M. and Stegun, I.A. (1964). *Handbook of mathematical functions*. Applied Mathematics Series volume 55, National Bureau of Standards, Washington. (reprinted 1968 by Dover Publications, New York).

Fisher, N.I. (1993). *Statistical analysis of circular data*. Cambridge University Press. Cambridge.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). *Numerical recipes in Fortran. The art of scientific computing. Second edition*. Cambridge University Press. Cambridge.

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION   'SPECIALFUNCTION example 1', !t('(Modified) Bessel functions', \
          'function of order 0 and 1 can be compared with Abramowitz', \
          'and Stegun (1964) pp 390 and 416'), ' '  ; STYLE=meta,2(plain)
VARIATE   [VALUES=0, 0.5 ... 5.0] x
SPECIALFUNCTION X=x ; J0BESSEL=j0 ; J1BESSEL=j1 ; I0BESSEL=i0 ; I1BESSEL=i1
CALCULATE ei0, ei1 = EXP(-x)*i0,i1
PRINT     x,j0,j1,ei0,ei1 ; FIELD=5,4(15) ; DECIMALS=1,4(8)
CAPTION   'SPECIALFUNCTION example 2', !t('The A1 function can be compared', \
          'with Fisher (1993), pp 225'), ' ' ; STYLE=meta,2(plain)
VARIATE   [VALUES=0.05,0.1...1.0,1.5,2...10,20,30...100] x
SPECIALFUNCTION X=x ; A1=a1
SPECIALFUNCTION X=a1 ; A1INVERSE=copyx
CALCULATE maxreldiff = MAX(ABS(x-copyx)/x)
PRINT     x,copyx,a1 ; FIELD=12 ; DECIMALS=2,4,4
PRINT     maxreldiff ; FIELD=-10 ; DECI=2
```

SREPLACE

L.C.P. Keizer & J.T.N.M. Thissen

Replaces (or removes) substrings from each string of a text structure

contents previous next

Options

REMOVE = *text*	Text structure with substrings to be removed from each string of the OLDTEXT parameter; default a single space `' '`
REPLACE = *text*	Text structure with substrings to be replaced in the positions of the substrings of the REMOVE text structure; default `''`
CASE = *string token*	Whether lower and upper case letters are to be regarded as identical when removing substrings (significant, ignored); default significant
CHANGE = *string token*	Whether the first or all occurrences must be replaced in each string (first, global); default global

Parameters

OLDTEXT = *texts*	Text structure from which substrings will be replaced; must be set
NEWTEXT = *texts*	To save the modified text structure

Description

Procedure SREPLACE can be used to replace or remove substrings from the strings of the OLDTEXT parameter. The modified text structure can be saved by means of the NEWTEXT parameter, or if NEWTEXT is not set the OLDTEXT structure will be overwritten by the modified one. The substrings to be removed from each string of OLDTEXT can be specified by the REMOVE option and the substrings to be replaced by the REPLACE option. If the REPLACE option is not set the substrings of REMOVE are removed and not replaced. The lengths of the REMOVE and REPLACE structures should be the same. The first value of REMOVE is replaced by the first value of REPLACE and so on. There is one exception: a vector-valued REMOVE text can be combined with a single-valued REPLACE text. Then each value of REMOVE is replaced by the value of REPLACE. The default settings of REMOVE and REPLACE are such that all spaces are removed from the strings of the OLDTEXT text structure.

The CASE option specifies whether lower and upper case letters are regarded as identical when replacing substrings. The CHANGE option specifies whether only the first occurrence in each string must be replaced or whether all occurrences must be replaced.

Method

Directives TXPOSITION and CONCATENATE are used in a loop.

Action with RESTRICT

If the OLDTEXT parameter is restricted, the NEWTEXT parameter is restricted in the same way. Values in units excluded by the restriction are not altered. Restrictions on REMOVE and REPLACE are ignored.

References

None**.**

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION   'SREPLACE example' ; STYLE=meta
TEXT      text ; VALUES=!t('Drs. Paul Keizer', 'Ir. Jac Thissen')
SREPLACE  text ; new
PRINT     text, new ; FIELD=20,23
SREPLACE  [CHANGE=first] text ; new
PRINT     text, new ; FIELD=20,23
SREPLACE  [REMOVE=!t(Paul, Jac) ; REPLACE=!t('L.C.P.', 'J.T.N.M.')] text ; new
PRINT     text, new ; FIELD=20,23
SREPLACE  [REMOVE=!t('drs. ', 'ir. ') ; CASE=ignore] text ; new
PRINT     text, new ; FIELD=20,23
SREPLACE  [REMOVE=!t(a,e,i,u)] text ; new
PRINT     text, new ; FIELD=20,23
SREPLACE  [REMOVE=!t(r,s) ; CHANGE=first] text ; new
PRINT     text, new ; FIELD=20,23
```

SUMMARIZE

J.C.M. Withagen

Prints summary statistics for variates

contents previous next

Options

PRINT = *string tokens*	What characteristics to print (mean, sd, %cv, median, min, max, nmv, nvalues, quantiles); default mean, sd, median, nmv, nvalues
PROPORTIONS = *numbers*	Proportions at which to calculate quantiles; default .10, .25, .50, .75, .90
REPRESENTATION = *string token*	Representation of values of summary statistics (exponential, standard); default exponential

Parameters

DATA = *variates*	Data to summarize; must be set

Description

Procedure SUMMARIZE calculates summary statistics for values stored in a variate as specified by the DATA parameter. The statistics to be calculated are indicated by the PRINT option. The summary is printed in a table with variate identifiers as rows and names of the summary statistics as columns. If PRINT=quantiles quantiles are calculated at the proportions specified by the PROPORTIONS option and printed in a separate table. By default values are presented in E-format. They can be presented in standard output format by the setting the REPRESENTATION option to standard.

Method

The procedure uses standard GenStat directives.

Action with RESTRICT

Any restriction on the data will be applied to all calculations.

References

None.

Procedures Used

None.

Similar procedures

DESCRIBE saves and/or prints summary statistics for variates, but in a different format.

Example

```
CAPTION   'SUMMARIZE example' ; STYLE=meta
CALCULATE data[1...5] = URAND(50697,4(0) ; 100)
SUMMARIZE [PRINT=#,quantiles ; REPRESENTATION=standard] data[]
```

SYM2VARIATE

J.T.N.M. Thissen

Copies a symmetric matrix to a variate along with row and column information

contents previous next

Options

`METHOD` = *string token*	Which elements of the symmetric matrix to copy (`offdiagonal`, `all`); default `offdiagonal`

Parameters

`SYMMETRICMATRIX` = *symmetric matrices*	Symmetric matrix to copy; must be set
`VARIATE` = *variates*	To save the values of the symmetric matrix; must be set
`RLEVELS` = *variates*	To save the row levels
`CLEVELS` = *variates*	To save the column levels
`RLABELS` = *texts*	To save the row labels
`CLABELS` = *texts*	To save the column labels
`RNUMBERS` = *variates*	To save the row ordinal numbers
`CNUMBERS` = *variates*	To save the column ordinal numbers

Description

Procedure `SYM2VARIATE` copies the elements of a symmetric matrix into a variate. All elements are copied by employing `METHOD=all`, while the default setting `METHOD=offdiagonal` only copies the off-diagonal elements. The values are saved in the `VARIATE` parameter. The row and column information can be saved in 3 ways:

1. If the `ROWS` option of the symmetric matrix is set to a text structure, the `RLABELS` and `CLABELS` parameters contain the relevant elements of this text and the `RLEVELS` and `CLEVELS` contain the numbers 1 till the dimension of the symmetric matrix.
2. If the `ROWS` option of the symmetric matrix is set to a variate, the `RLABELS` and `CLABELS` parameters contain the relevant elements of this variate as strings and the `RLEVELS` and `CLEVELS` contain the same values as numbers in a variate.
3. If the `ROWS` option of the symmetric matrix is set to a scalar, the `RLABELS` and `CLABELS` parameters contain the numbers 1 till this scalar as strings and the `RLEVELS` and `CLEVELS` contain the numbers 1 till the dimension of the symmetric matrix.

The `RNUMBERS` and `CNUMBERS` parameters always contain the numbers 1 till the dimension of the symmetric matrix.

Method

The procedure uses standard GenStat directives.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

None.

Similar procedures

None.

Example

```
CAPTION     'SYM2VARIATE examples' ; STYLE=meta
SYMMETRIC   [ROWS=!t(a,b,c,d,e) ; VALUES=1...15] sym
PRINT       sym ; FIELD=8 ; DECIMALS=0
SYM2VARIATE sym ; VARIATE=v ; \
            RLABELS=rlab ; CLABELS=clab ; RLEVELS=rlev ; CLEVELS=clev ; \
            RNUMBERS=rnum ; CNUMBERS=cnum
PRINT       v,rlab,clab,rlev,clev,rnum,cnum ; FIELD=8 ; DECIMALS=0
CAPTION     'SYM2VARIATE example 2' ; STYLE=minor
SYMMETRIC   [ROWS=!t(a,b,c,d,e) ; VALUES=1...15] sym
PRINT       sym ; FIELD=8 ; DECIMALS=0
SYM2VARIATE [METHOD=all] sym ; VARIATE=v ; \
            RLABELS=rlab ; CLABELS=clab ; RLEVELS=rlev ; CLEVELS=clev ; \
            RNUMBERS=rnum ; CNUMBERS=cnum
PRINT       v,rlab,clab,rlev,clev,rnum,cnum ; FIELD=8 ; DECIMALS=0
CAPTION     'SYM2VARIATE example 3' ; STYLE=minor
SYMMETRIC   [ROWS=!(1,4,9,16,25) ; VALUES=1...15] sym
PRINT       sym ; FIELD=8 ; DECIMALS=0
SYM2VARIATE [METHOD=all] sym ; VARIATE=v ; \
            RLABELS=rlab ; CLABELS=clab ; RLEVELS=rlev ; CLEVELS=clev ; \
            RNUMBERS=rnum ; CNUMBERS=cnum
PRINT       v,rlab,clab,rlev,clev,rnum,cnum ; FIELD=8 ; DECIMALS=0
CAPTION     'SYM2VARIATE example 4' ; STYLE=minor
SYMMETRIC   [ROWS=5 ; VALUES=1...15] sym
PRINT       sym ; FIELD=8 ; DECIMALS=0
SYM2VARIATE [METHOD=all] sym ; VARIATE=v ; \
            RLABELS=rlab ; CLABELS=clab ; RLEVELS=rlev ; CLEVELS=clev ; \
            RNUMBERS=rnum ; CNUMBERS=cnum
PRINT       v,rlab,clab,rlev,clev,rnum,cnum ; FIELD=8 ; DECIMALS=0
```

TPOWER

P.W. Goedhart & M.J.W. Jansen

Calculates the power of Student's t-test and plots power curves

contents previous next

Options

`PRINT` = *string tokens*	What to print (`description`, `power`); default `description`, `power`
`POWERCURVE` = *string tokens*	Power curve to plot (`none`, `effect`, `nreplicates`); default `none`
`DESIGN` = *string token*	Designed experiment for which power must be calculated (`random`, `block`, `latinsquare`, `onesample`, `general`); default `random`
`VARIANCE` = *scalar*	Estimate of unit variance; default 1
`NTREATMENTS` = *scalar*	Number of treatments in a designed experiment; default 2
`METHOD` = *string token*	Type of test required (`onesided`, `towsided`); default `twosided`
`PROBABILITY` = *scalar*	Significance level at which the effect is required to be detected; default 0.05
`ADFCONSTANT` = *scalar*	Constant for residual degrees of freedom of a general design; default `*`
`BDFCONSTANT` = *scalar*	Constant for residual degrees of freedom of a general design; default `*`
`CVAREFFECT` = *scalar*	Constant for the variance of the effect for a general design; default `*`
`DVARCONSTANT` = *scalar*	Constant for the variance of the effect for a general design; default 0
`ANNOTATION` = *string tokens*	Defines the annotation of the power curves (`description`, `curves`, `lines`); default `description`, `curves`, `lines`
`LINESATPOWER` = *scalars*	Power values for which horizontal lines are added to the plotted power curves, along with vertical lines at intersections with the power curves; default 0.9
`WINDOW` = *scalars*	Window numbers for the power curves for effect and/or nreplicates respectively; default `*` uses a full screen window
`SCREEN` = *string token*	Whether to clear the screen before plotting both power curves or to continue plotting on the old screen (`clear`, `keep`); default `clear`
`TITLE` = *texts*	General titles of the power curves for effect and/or nreplicates respectively; default `*` uses default titles
`YTITLE` = *texts*	Titles for the y-axis of the power curves for effect and/or nreplicates respectively; default 'Power'
`XTITLE` = *texts*	Titles for the x-axis of the power curves for effect and/or nreplicates respectively; default `*` uses default titles

Parameters

`EFFECTS` = *variates* or *scalars*	Effects for which the power has to be calculated; must be set
`NREPLICATES` = *variates* or *scalars*	Number of replicates for which the power must be calculated; must be set
`POWER` = *tables*	Saves the power of Student's t-test

Description

Procedure `TPOWER` can be used to calculate the power of Student's t-test for some standard experimental designs and also in the general case. The `EFFECTS` and `NREPLICATES` parameters specify the effects and the number of replicates for which the power must be calculated. The procedure calculates the power for all combinations of the values in `EFFECTS` and `NREPLICATES`. The two-way table with power values can be printed by setting the `PRINT` option, or saved by means of the `POWER` parameter. Graphs with power curves can be requested by setting the `POWERCURVE` option.

The `DESIGN` option specifies which experimental design is used. The following designs are available:

- `random` a completely randomized experiment with equally replicated treatments; the values in `NREPLICATES` are the number of replications of each treatment;
- `block` a randomized complete block experiment; the values in `NREPLICATES` are the number of blocks;

- `latinsquare` a set of latin squares; the values in `NREPLICATES` are the number of squares;
- `onesample` a random sample with only one treatment; the values in `NREPLICATES` are the sample sizes;
- `general` general design; see below.

The `VARIANCE` option must be set to the variability of units with the same treatment; this will in general be set to the residual mean square of a previous analysis of the same kind. The `NTREATMENTS` option specifies the number of treatments for the `random`, `block` and `latinsquare` designs. By default the number of treatments is set to two, which gives a two-sample t-test for `DESIGN=random`, a pairwise t-test for `DESIGN=block` and 2 x 2 latin squares for `DESIGN=latinsquare`. The `METHOD` option specifies whether the t-test is one-sided or two-sided. For a one-sided test the null hypothesis is that the effect is less than or equal to zero. For a two-sided test the null hypothesis is that the effect equals zero. The significance level at which an effect is required to be detected must be specified by the `PROBABILITY` option.

The power for more general setups can be obtained by specifying `DESIGN=general`, in which case the `ADFCONSTANT`, `BDFCONSTANT`, `CVAREFFECT` and `DVARCONSTANT` options define the properties of the t-test. The degrees of freedom (dfresidual) of the residual variance, and the variance (vareffect) of the estimated effects are then defined by:

dfresidual = `ADFCONSTANT * NREPLICATES – BDFCONSTANT.`

vareffect = `DVARCONSTANT + VAREFFECT * VARIANCE / NREPLICATES`

For the standard designs the baseline variance `DVARCONSTANT` equals zero and the other constants are defined as follows, where *k* is the number of treatments:

- `random`	`ADFCONSTANT`=*k*;	`BDFCONSTANT`=*k*;	`CVAREFFECT`=2
- `block`	`ADFCONSTANT`=*k*–1;	`BDFCONSTANT`=*k*–1;	`CVAREFFECT`=2
- `latinsquare`	`ADFCONSTANT`=(*k*–1)*(*k*–1);	`BDFCONSTANT`=*k*–1;	`CVAREFFECT`=2/*k*
- `onesample`	`ADFCONSTANT`=1;	`BDFCONSTANT`=1;	`CVAREFFECT`=1

Note that the interpretation of the `NREPLICATES` values depends on the settings of the design constants. The setting of the `NTREATMENTS` option is discarded for the general case. The example program lists a number of general designs.

The `POWERCURVE` option can be set to request two types of powercurves. This is only useful for small number of values in the `EFFECTS` and `NREPLICATES` parameters. `POWERCURVE=effect` produces a graph with power curves as a function of the required effect, separately for every value in the `NREPLICATES` parameter. The efffect ranges from the minimum to the maximum of the `EFFECTS` parameter, with a zero effect added. `POWERCURVE=nreplicates` plots power curves as a function of the number of replicates, now separately for every positive value in the `EFFECTS` parameter. The nreplicates ranges from the minumum to the maximum of the `NREPLICATES` parameter. The `ANNOTATION` option defines the annotation of the powercurves: `description` displays an informative box, `curves` adds annotation to the curves and `lines` displays horizontal (and accompanying vertical lines) at power values specified by the `LINESATPOWER` option. Window settings and titles for both graphs, i.e. for effect and/or nreplicates power curves respectively, can be specified by setting options `WINDOW`, `SCREEN`, `TITLE`, `YTITLE` and `XTITLE`. These options can be set to two values, the first setting is for the `effect` curve, the second for the `nreplicates` curve. Pen number 1 is used for the curves, pen 33 for the description and the almost black pen 60 for annotation of the curve. Pen numbers 2,3… are used for the lines as specified by the `LINESATPOWER` option. The axis labels and titles and the general title can be controled by the default negative pen numbers.

Method

The `TPOWER` procedure employs the non-central t-distribution. The basic calculations are as follows:

```
CALCULATE dfresidual = ADFCONSTANT*NREPLICATES – BDFCONSTANT
CALCULATE vareffect = DVARCONSTANT + CVAREFFECT*VARIANCE/NREPLICATES
CALCULATE tvalue = EDT(PROBABILITY ; dfresidual)
CALCULATE noncentral = EFFECTS/SQRT(vareffect)
IF (METHOD.EQS.’onesided’)
    CALCULATE POWER = CUT(tvalue ; df ; noncentral)
  ELSE
```

```
        CALCULATE POWER = CUT( ABS(tvalue) ; df ; ABS(noncentral)) + \
                          CLT(-ABS(tvalue) ; df ; ABS(noncentral))
    ENDIF
```

Notably for small number of replicates the calculation of the tail probabilities of the non-central t-distribution may not converge, and the CUT and CLT functions return a zero value. Non convergence results in a fault code CA 58, in which case all zero power values are replaced by missing values. Power curves with one or more missing values, again due to non convergence, are not plotted.

Action with RESTRICT

Restrictions on the EFFECTS and NREPLICATES parameters are taken into account.

References

None**.**

Procedures Used

DECIMALS, FTEXT, SUBSET and VEQUATE.

Similar Procedures

Procedures ASAMPLESIZE, APOWER, RPOWER, XOPOWER, STTETS, SBNTEST, SCORRELATION, SSIGNTEST, SMANNWHITNEY, SMCNEMAR and SLCONCORDANCE calculate power and sample sizes for various statistical tests.

Example

```
CAPTION   'TPOWER example 1', !t('A completely randomized experiment', \
          'with two treatments'), ' ' ; STYLE=meta,2(plain)
VARIATE   [VALUES=0.50, 0.75 ... 1.50] effect
VARIATE   [VALUES=3...20] nrep
TPOWER    effect ; nrep
TPOWER    [PRINT=* ; POWERCURVE=effect,nreplicates] effect ; !(5,10,20,30)
CAPTION   'TPOWER example 2', !t('A balanced incomplete block experiment', \
          'with three treatments. Each replicate consists of three blocks'), \
          ' ' ; STYLE=meta,2(plain)
FACTOR    [LEVELS=3] treatment, block ; !(1,2,1,3,2,3), !(1,1,2,2,3,3)
BLOCK     block
TREATMENT treatment
ANOVA     [PRINT=*] URAND(7474 ; NVALUES(treatment))
AKEEP     treatment ; EFFICIENCY=efficiency ; REPLICATION=replication
SCALAR    a,b ; 3, 2
SCALAR    c ; 2/(replication*efficiency)
TPOWER    [DESIGN=general ; ADF=a ; BDF=b ; CVAR=c] effect ; nrep
CAPTION   'TPOWER example 3', !t('A block experiment in which 6 treatments', \
          'are all compared with an added control which is replicated four', \
          'times in each block. The number of replicates equals the number', \
          'of blocks'), ' ' ; STYLE=meta,2(plain)
SCALAR    a, b, c ; 9, 6, 1.25
TPOWER    [DESIGN=general ; ADF=a ; BDF=b ; CVAR=c] effect ; nrep
CAPTION   'TPOWER example 4', !t('Testing beta=0 in simple linear', \
          'regression. The number of replicates is the number of times', \
          'the regressor x is repeated'), ' ' ; STYLE=meta,2(plain)
VARIATE   x, y ; !(0,1,2,3)
MODEL     [DISPERSION=1] y
FIT       [PRINT=*] x
RKEEP     SE=sebeta
SCALAR    a ; NVALUES(x)
SCALAR    b ; 2
SCALAR    c ; sebeta$[2]**2
TPOWER    [DESIGN=general ; ADF=a ; BDF=b ; CVAR=c ; VARIANCE=4 ; \
          METHOD=onesided ; PROBABILITY=0.01] effect ; nrep
```

TSQUEEZE

L.C.P. Keizer & J.T.N.M. Thissen

Squeezes a table to fewer levels of the classifying factors

Options

PRINT = *string token*	What information to print (newtable); default *
MINIMUM = *scalar*	Minimum value for the values of TABLE ; default 0
MAXIMUM = *scalar*	Maximum value for the values of TABLE ; default *

Parameters

TABLE = *tables*	Table without margins which should be squeezed; must be set
NEWTABLE = *tables*	To save the squeezed TABLE
SIMILARTABLES = *pointers*	Tables, which should be squeezed in the same way as TABLE
NEWSIMILARTABLES = *pointers*	To save the squeezed SIMILARTABLES

Description

Procedure TSQUEEZE can be used to display or save a table with fewer levels of the classifying factors. The table should not have margins. By default those levels of the classifying factors of the TABLE parameter are removed that only have missing values or values less than or equal to zero in all corresponding cells. The MINIMUM and MAXIMUM options can be used to reduce or expand the range of values. Levels of factors with all values in the TABLE parameter less than or equal to MINIMUM or all values greater than MAXIMUM are then removed.

The squeezed table can be saved in the NEWTABLE parameter. When there are more tables that must be squeezed in the same way as TABLE they can be set by the SIMILARTABLES parameter. The new squeezed tables can be saved then in the NEWSIMILARTABLES pointer. The setting PRINT=newtable prints the squeezed table. The names of the classifying factors of the squeezed tables are formed from those of the original tables; For example, if the original table has a classifying factor “name”, the squeezed table has a classifying factor “_name_”.

Method

The COMBINE directive is used to squeeze the tables.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

SUBSET and FPOINTER.

Similar Procedures

None.

Example

```
CAPTION   'TSQUEEZE example' ; STYLE=meta
FACTOR    [NVALUES=100 ; LEVELS=4] f1
FACTOR    [NVALUES=100 ; LEVELS=5] f2
GENERATE  f1, 5, f2
CALCULATE initialize = URAND(90124 ; 1)
CALCULATE x = GRNORMAL(100 ; 10 ; 8)
TABULATE  [CLASSIFICATION=f1, f2] x ; MEAN=mean ; VARIANCE=var
PRINT     [SERIAL=yes] mean, var
TSQUEEZE  [MINIMUM=11] TABLE=mean ; NEWTABLE=newmean ;\
          SIMILAR=!p(var) ; NEWSIMILAR=!p(newvar)
PRINT     [SERIAL=yes] newmean, newvar
V2TABLE   [CLASSIFICATION=newf1,newf2] newmean,newvar ; VARIATE=vmean,vvar
PRINT     newf1, newf2, vmean, vvar
```

V2TABLE

P.W. Goedhart

Forms a variate and a set of classifying factors from a table

contents previous next

Options

`CLASSIFICATION` = *factors*	To save the (ordered) classifying set of the table; default `*`
`MODIFY` = *string token*	Whether to modify the classifying factors of the table (`no`, `yes`); `MODIFY` is only relevant when the `CLASSIFICATION` option is unset; default `no`

Parameters

`TABLE` = *tables*	Tables to be copied
`VARIATE` = *variates*	To save the body of each table

Description

Procedure `V2TABLE` can be used to store the body of a table in a variate and obtain a set of factors to represent the way in which the data are arranged in the table. These factors are then of the same length as the newly formed variate and classify the variate in the same way as in the table. Margins of the table are ignored.

The tables to be copied are specified by the `TABLE` parameter, the variates to receive the body of the tables must be specified by means the `VARIATE` parameter. The tables should have the same classifying factors. The `DECIMALS` and `EXTRA` attributes of the tables are transferred to the variates.

The `CLASSIFICATION` option can be used to obtain the (ordered) classifying factors of the first table. The newly formed factors have the same attributes as the classifying factors of the table. Alternatively, if the `CLASSIFICATION` option is unset or set to `*`, the option setting `MODIFY=yes` can be used to shorten the classifying factors of the table so that they classify the newly formed variate.

Note that the order in which the factors are obtained can be unexpected for implicitly declared tables. To avoid confusion, the list of factors as specified by the `CLASSIFICATION` option, is compared with the list of ordered classifying factors of the table. If one or more factors in the ordered classifying list are in the `CLASSIFICATION` list, there position in these lists should be the same. If this is not the case a fault is generated. For example, the following lines will produce a fault message:

```
TABLE   [CLASSIFICATION=f1, f2 ; VALUES=1,2,3,4] table
V2TABLE [CLASSIFICATION=f2, f1] table ; variate
```

Method

To ensure that all tables have the same ordered classifying set, the tables are first copied to tables with the ordered classifying set of the first table. Margins of the table are then deleted by the `MARGIN` directive and the tables are equated to variates. The initial declarations of the new factors are done with `DUPLICATE`. Factor values are produced by `GENERATE`.

Action with RESTRICT

Not relevant.

References

None.

Procedures Used

`V2TABLE` calls the subsidiary procedure `%V2TABLECHECK` which checks that the tables have the same classifying factors.

Similar Procedures

VTABLE from the official GenStat Procedure Library can be used for tables with different classification sets.

Example

```
CAPTION   'V2TABLE example' ; STYLE=meta
FACTOR    [LEVELS=4 ; VALUES=12(1),15(2),13(3),14(4)] Block
FACTOR    [LABELS=!T('Nitrogen+','Nitrogen0','Nitrogen-') ; \
          VALUES=4(1,2,3), 5(1,2,3), 4(1,2,3),3, 5(1,2),4(3)] Diet
VARIATE   [NVALUES=54] Milk
READ      Milk ; DECIMALS=1
   312 330 300 287        294 291 303 289        275 282 281 290
   278 284 281 263 289    294 283 281 274 298    264 270 288 285 248
   290 256 265 243        270 261 256 279        253 259 268 240 242
   276 243 233 238 259    245 241 227 255 222    235 227 227 247 :
TABULATE  [CLASSIFICATION=Block,Diet] Milk ; MEAN=MeanMilk ; \
          NOBSERVATION=NobsMilk
V2TABLE   [CLASSIFICATION=NewBlock, NewDiet] TABLE=NobsMilk ; VARIATE=Nobs
PRINT     NobsMilk
PRINT     NewBlock,NewDiet, Nobs
V2TABLE   [MODIFY=yes] TABLE=MeanMilk ; VARIATE=Milk
PRINT     MeanMilk
PRINT     Block, Diet, Milk
```

VSEARCH

P.W. Goedhart

Helps search through models for a generalized linear mixed model (GLMM)

contents previous next

Options

`PRINT` = *string tokens*	What output to display (`terms`, `details`, `changes`, `model`, `components`, `waldtests`); default `details`, `changes`, `model`, `components`, `waldtests`
`DISTRIBUTION` = *string token*	Error distribution (`normal`, `binomial`, `poisson`, `gamma`, `negativebinomial`); default `normal`
`LINK` = *string token*	Link function (`identity`, `logarithm`, `logit`, `reciprocal`, `probit`, `complementaryloglog`, `logratio`); default `*` gives the canonical link
`DISPERSION` = *scalar*	Value at which to fix the residual variance, if missing the variance is estimated; default 1
`RANDOM` = *formula*	Random model excluding bottom stratum; this must be set
`FREE` = *formula*	Model formula specifying the candidate fixed model terms; this must be set
`STARTFREE` = *formula*	Model formula specifying the candidate fixed model terms with which to start the stepwise procedure; default `*` starts with an empty model
`FORCED` = *formula*	Fixed model formula to include in each model; default `*`
`ADDPROBABILITY` = *scalar*	p-value of significance test for adding candidate model terms; default 0.05
`DROPPROBABILITY` = *scalar*	p-value of significance test for dropping of candidate model terms; default 0.05
`MAXSTEPS` = *scalar*	Number of times the main stepwise loop is executed; default 1000
`REPEAT` = *string tokens*	Whether to repeat dropping or adding model terms within the main stepwise loop (`drop`, `add`); default `*` does not repeat either
`PMETHOD` = *string token*	How p values are calculated (`chisquared`, `fdistribution`); default `fdistribution`
`SELECTEDMODEL` = *formula*	Saves the selected model
`GLMMINITIAL` = *string token*	Whether to use initial fitted values from a previous model fit in the model sequence or not (`yes`, `no`); default `yes`
`CONSTANT` = *string token*	Whether to estimate or omit the constant term in the fixed model (`omit`, `estimate`); default `estimate`
`FACTORIAL` = *scalar*	Limit for expansion of fixed model terms; default 3
`MAXCYCLE` = *scalar*	Maximum number of iterations of the GLMM algorithm; default 20
`TOLERANCE` = *scalar*	Convergence criterion for the iterative GLMM algorithm; default 0.0001
`FMETHOD` = *string token*	Specifies fitting method (`all`, `fixed`): `all` indicates the method of Schall (1991); `fixed` indicates the marginal method of Breslow & Clayton (1993) ; default `all`
`OFFSET` = *variate*	Offset variate to be included in the fixed model; default *
`CADJUST` = *string token*	What adjustment to make to covariates for the REML analysis (`mean`, `none`); default `mean`
`AGGREGATION` = *scalar*	Fixed parameter for negative binomial distribution (parameter k as in variance function var = $\mu + \mu^2/k$); default 1
`KLOGRATIO` = *scalar*	Parameter *k* for logratio link, in form $\log(\mu / (\mu + k))$; default as set in `AGGREGATION` option
`OWNDIST` = *text*	For non-standard distributions: text specifying the variance function to be used with dummy variable `DUM`, e.g. `OWNDIST='DUM'`
`OWNLINK` = *text*	For non-standard link functions: text specifying 3 functions using dummy variable `DUM` - the link function, its inverse and its derivative,

	e.g. OWNLINK = !T('log(DUM)','exp(DUM)','1/DUM')
CDEFINITIONS = *text*	Statements to execute to define correlation models; default * i.e. none
CVECTORS = *pointer*	Data structures involved in the correlation models
WORKSPACE = *scalar*	Number of blocks of internal memory to be set up for use by the REML algorithm; default 1

Parameters

Y = *variates*	Response variates
NBINOMIAL = *scalars* or *variates*	Number of binomial trials for each unit (must be set if DISTRIBUTION=binomial)

Description

VSEARCH can be used to perform stepwise selection of fixed model terms in a generalized linear mixed model by employing the GLMM procedure. Most of the options and parameters of the VSEARCH procedure originate form the GLMM procedure. The options specific to VSEARCH are FREE, STARTFREE, FORCED, ADDPROBABILITY, DROPPROBABILITY, MAXSTEPS, REPEAT, PMETHOD, SELECTEDMODEL and GLMMINITIAL.

The FREE option specifies the candidate model terms. These may include variates, factors and interactions. It is sometimes desirable to include specific terms in each model. Such terms may be specified by means of the FORCED option. If the FREE formula specifies a main effects model, i.e. a model without interactions, all main effects are the candidate terms. When the FREE formula contains interactions, first all terms marginal to an interaction are dropped from the FREE formula and are added to the FORCED formula. This ensures that the principle of marginality is never violated when the candidate terms are fitted in turn. The STARTFREE option specifies the candidate model terms with which to start the selection procedure.

Each iteration of the stepwise procedure consists of two parts. In the first part it is tested whether any of the current model terms can be dropped. This is done by fitting the current model and obtaining significance tests for all candidate terms in the current model. If the maximum p-value of these significance tests exceeds the value of the DROPPROBABILITY option, then the corresponding model term is dropped from the current model. In the second part it is tested whether any of the candidate model terms which are not in the current model can be added. This is done by adding these terms to the current model and obtaining a significance test. Note that these are single additions to the current model. If the minimum p-value of these tests is smaller than the value of the ADDPROBABILITY option, then the corresponding model term is added to the current model. After these two parts, the next iteration of the stepwise procedure starts. By specifying REPEAT=drop the first part itself is executed in a loop until no further terms can be dropped; after this loop the second part is executed. Likewise REPEAT=add involves repeated addition of model terms in the second part before any terms can be dropped.

Note that forward selection, i.e. no terms are ever dropped from the model, can be requested by setting the DROPPROBABILITY option to 1. Likewise backward elimination, i.e. no terms are ever added to the model, is obtained by specifying ADDPROBABILITY=0. In the latter case the STARTFREE and FREE options should be set to the same model formula.

The PMETHOD option controls how p-values are calculated for the significance tests. The significance test is always based on the Wald statistic. PMETHOD=chisquared calculates the p-value according to the chi-squared distribution. Alternatively PMETHOD=fdistribution employs the F statistic, which is the Wald statistic divided by its degrees of freedom. The p-value is then calculated with the F distribution with approximate denominator degrees of freedom as obtained by setting FMETHOD=auto of VKEEP. In case the denominator degrees of freedom is not available the chi-squared distribution is used. Note that in both cases the F statistic is printed.

The GLMMINITIAL option controls whether the INITIALVALUES parameter of the GLMM procedure is set to fitted values from a previous model fit. Setting this option to yes reduces running time but can occasionally result in a failure of the GLMM algorithm to reach convergence

Output is controlled by the PRINT option. The changes setting lists all the changes made to the model, while the details setting prints the sorted significance tests in each step of the stepwise procedure. Note that estimates and standard errors are printed for effects with one degree of freedom; these should be interpreted with care as they are specific for the model that is current. The `terms` setting prints the forced and free terms which are used in the stepwise selection. The other `PRINT` settings, i.e. `model`, `components` and `waldtests`, can be used to display results for the selected model. `VDISPLAY` and `VKEEP` can also be used after procedure `VSEARCH` to redisplay or store other results for the selected model.

The response variate is specified using the `Y` parameter. The `NBINOMIAL` parameter must be set when `DISTRIBUTION=binomial` to specify the total number of trials on each unit, as a variate if the number varies from unit to unit or as a scalar if it is constant over all the units.

All other options are directly passed to the `GLMM` procedure; consult the description of `GLMM` for a full explanation. The `DISTRIBUTION` option sets the error distribution; the default is to assume a normal distribution but the binomial, poisson, gamma and negative-binomial distributions are also available. The link can be set using the `LINK` option; the default takes the canonical link. Other distributions and links can be employed by setting the `OWNDIST` and `OWNLINK` options. The `AGGREGATION` option supplies the aggregation parameter for the negative-binomial distribution, which is 1 by default. The `KLOGRATIO` option supplies the parameter *k* to be used in the logratio link, and takes its default from `AGGREGATION`.

The random model is specified by the `RANDOM` option. The dispersion parameter is assumed to be 1 unless otherwise specified by the `DISPERSION` option. Setting `DISPERSION=*` requests that the dispersion parameter be estimated. It is also possible to define correlation models on the random terms, although the results should be used with caution as their properties are not yet well understood. To do this, you should set the `CDEFINITIONS` and `CVECTORS` options as is explained in the description of the `GLMM` procedure.

The number of identifiers in free and forced terms can be limited using the `FACTORIAL` option. By default, a constant term is included in the model; this can be suppressed by setting option `CONSTANT=omit`. An offset can be included in the linear predictor by setting option `OFFSET`. By default all covariates are centred by subtracting their means, weighted according to the iterative weights of the generalized linear model. You can set option `CADJUST=none` to request that the uncentred covariates are used instead.

The `FMETHOD` option specifies the method used to form the fitted values and therefore determines the fitting method to be used. The default setting `all` specifies the penalized quasi likelihood method which is a subject specific model (Schall, 1991), while setting `fixed` requestst the marginal quasi likelihood method; see Breslow & Clayton (1993). Some control over the iterative GLMM algorithm is provided by option `MAXCYCLE` which sets the maximum number of iterations (default 20), and by option `TOLERANCE` which specifies the criterion for determining convergence of the algorithm (default 0.0001). The `WORKSPACE` option (default 1) specifies the number of blocks of internal memory to be allocated by the `REML` directive.

Method

`VSEARCH` repeatedly calls the `GLMM` procedure to obtain significance tests for fixed terms to drop or add. Any warning or message diagnostics produced by the `GLMM` procedure are suppressed, except when fitting the selected model. The stepwise selection process can result in an indefinite loop, e.g. when a term has a p-value of 0.07 with `ADDPROBABILITY`=0.10 and `DROPPROBABILITY`=0.05. This is detected by the procedure in which case the main loop is exited.

Action with RESTRICT

Only the response variate can be restricted. The analysis is restricted accordingly. Identifiers in the fixed and random formulae must not be restricted and must not contain missing values.

References

Breslow, N.E. & Clayton, D.G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 421, 9-25.

Schall, R. (1991) Estimation in generalized linear models with random effects. *Biometrika*, **78**, 719-727.

Procedures Used

The subsidiary procedure _RSEARCHCHECK checks all the identifiers which are involved in the model. The generalized linear mixed models are fitted using the GLMM procedure and test statistics are obtained with the VWALD procedure.

Similar Procedures

RSEARCH helps search through models for a regression or generalized linear model. RSELECT selects best predictor variables in ordinary linear regression.

Example

```
CAPTION   'VSEARCH example' ; STYLE=meta
BIOMETRIS 'VSEARCH' ; DATA=DataVsearch
EXECUTE   DataVsearch
POINTER   [VALUES=age, xero, cosine, sine, female, height, stunted] free
VSEARCH   [DISTRIBUTION=binomial ; RANDOM=child ; FREE=free[]] \
          resp ; NBINOMIAL=1
VSEARCH   [DISTRIBUTION=binomial ; RANDOM=child ; FREE=free[] ; \
          STARTFREE=free[] ; REPEAT=drop] resp ; NBINOMIAL=1
VSEARCH   [DISTRIBUTION=binomial ; RANDOM=child ; FACTORIAL=2 ; \
          FREE=free[] + age*cosine*sine*height*stunted] resp ; NBINOMIAL=1
```

VWALD

P.W. Goedhart, W.G. Buist & B. Engel

Saves non-hierarchical Wald tests for fixed terms in a REML analysis

Options

`PRINT` = *string token*	Whether to print the test statistics and associated probabilities and degrees of freedom (`test`); default `test`
`SORT` = *string token*	Whether to sort the results of the tests into ascending order according to the probabilities (`yes`, `no`); default `no`
`PMETHOD` = *string token*	Controls which distribution to use for calculating p values (`chisquared`, `fdistribution`); default `fdistribution`
`WMETHOD` = *string token*	Controls which Wald statistics are saved (`add`, `drop`); default `drop`

Parameters

`RESULTS` = *pointer*	Pointer to save the tested terms and the test results
`ADDLAST` = *pointer*	Pointer to save results of the last model term in the `FIXED` formula as specified by means of `VCOMPONENTS`
`DROPFIRST` = *pointer*	Pointer to save results of the model term with the largest P value

Description

A linear mixed model can be analysed by Restricted Maximum Likelihood (REML), as explained by e.g. Engel (1990). The `REML` directive implements REML, and significnace (Wald) tests for fixed effects can be obtained with `PRINT=waldtest`. Two sets of Wald tests are then printed; these are named "Sequentially adding terms to fixed model" and "Dropping individual terms from full fixed model". This procedure can be used to save (and print) the latter Wald tests. Only tests for terms which are not marginal to a higher order interaction are produced.

The `PMETHOD` option controls how p-values are calculated for the significance tests. The significance test is always based on the Wald statistic. `PMETHOD=chisquared` calculates the p-value according to the chi-squared distribution. Alternatively `PMETHOD=fdistribution` employs the F statistic, which is the Wald statistic divided by its degrees of freedom. The p-value is then calculated by means of the F distribution with approximate denominator degrees of freedom as obtained by setting `FMETHOD=auto` of `VKEEP`. In case the denominator degrees of freedom is not available the chi-squared distribution is used. Note that in both cases the F statistic is printed. The denominator degrees of freedom is set to missing when the chi-squared distribution is used. The `WMETHOD` option controls whether the test statistics are for all terms which are added sequentially to the model, or only for those terms that can be dropped from the model. In the former case the principle of marginality might be violated.

The `PRINT` option controls the output of `VWALD`. For each term the F statistic (Fvalue) is printed along with its numerator (Ndf) and denominator (Ddf) degrees of freedom and the associated P value (Pvalue). For tems with a single degree of freedom the estimated effect and its standard error are also printed. The `SORT` option can be used to sort the results of the tests into ascending order according to the pvalue of the Wald tests. The `RESULTS` parameter can be used to save the tested terms and the test results. The `ADDLAST` and `DROPFIRST` parameters can be used to save results of specific terms of the `FIXED` model formula. `ADDLAST` and `DROPFIRST` can be used to implement model selection by means of forward selection, backward elimination or stepwise selection.

Method

The fixed model formula is broken up into individual terms and for each term the Wald statistic is basically calculated as follows:

```
VKEEP     [FMETHOD=auto] TERMS=term ; \
          WALD=wald ; FSTATISTIC=fstat ; NDF=ndf ; DDF=ddf
IF (PMETHOD.EQS.'CHISQUARED') .OR. (ddf.EQ.C('*'))
   CALCULATE ddf = missing
```

```
    CALCULATE pvalue = CUCHI(wald ; ndf)
  ELSE
    CALCULATE pvalue = CUF(fstat ; ndf ; ddf)
ENDIF
```

Note that only the F statistic is printed as the Wald statistic equals the F statistic times its (numerator) degrees of freedom. When the constant is omitted from the model it is included into the effects of the first term including a factor of the fixed model. The calculated statistic for this term can therefore be misleading and `VWALD` will print a warning message.

Action with RESTRICT

Not relevant.

References

Engel, B. (1990). The analysis of unbalanced linear models with variance components. *Statistica Neerlandica*, **44**, 195-219.

Procedures Used

None.

Similar Procedures

None.

Example

```
CAPTION     'VWALD example' ; STYLE=meta
BIOMETRIS   'VWALD' ; DATA=DataVwald
EXECUTE     DataVwald
VCOMPONENTS [FIXED=dose + sex + littersz] RANDOM=dam + pups
REML        [PRINT=components,waldtest] weight
VWALD
```

WEAVEVECTORS

L.C.P. Keizer & J.T.N.M. Thissen

Weaves two sets of vectors into a new set according to the first vector of both sets

contents previous -

Options

SORT = *string token*	Whether to sort the target vector (yes, no); default no
DIRECTION = *string token*	Order in which to sort the target vector (ascending, descending); default ascending

Parameters

FIRSTSET = *pointers*	First set of vectors to interweave; must be set
SECONDSET = *pointers*	Second set of vectors to interweave; must be set
COMBINATION = *pointers*	To save the combined sets of vectors; must be set

Description

Procedure WEAVEVECTORS can be used to weave two sets of vectors into a new set of vectors according to the first vector of both sets. WEAVEVECTORS is especially useful when combining two data sets with a common target vector. The two sets are specified by parameters FIRSTSET and SECONDSET. The type of the first vector in both sets must be the same, either a variate or a text. Each of the first vectors must have unique values. The weaving goes as follows: take all the elements of the first vector of FIRSTSET and add those elements of the first vector of SECONDSET not equal to any element of the first vector in FIRSTSET. The result of the weaving is a target vector according to which both sets of vectors are combined. The target vector is the first vector of the COMBINATION pointer with which the result must be saved. The other vectors of COMBINATION are then subsequently the other modified vectors of FIRSTSET and the other modified vectors of SECONDSET.

Option SORT can be used, in combination with the DIRECTION option, to sort the target vector in ascending or descending order. If SORT=no the elements of the target vector are the elements of the first vector in FIRSTSET supplemented by the elements of the first vector in SECONDSET not equal to the elements of the first vector in FIRSTSET.

Method

The weaving is done with directive EQUATE and proper specifications of the options OLDFORMAT and NEWFORMAT.

Action with RESTRICT

The vectors in FIRSTSET and SECONDSET must not be restricted.

References

None**.**

Procedures Used

None.

Similar Procedures

MATCHTARGET extracts units of a set of vectors according to a target vector. JOIN joins or merges two sets of vectors together, based on the values of sets of classifying keys.

Example

```
CAPTION   'WEAVEVECTORS example' ; STYLE=meta
TEXT      tvariety1, tvariety2
READ      [PRINT=data,errors] nr1, tvariety1, loc[1...3]
   1  Ritmo         10.5  10.7  10.8
   2  Hereward      11.6  12.1  12.2
   3  Vivant        10.4  10.7  10.8
   4  Bercy         11.1     *     *
   5  Versailles    10.6     *     *
   6  Arnaut        12.0  11.7  11.4
   7  Tambor        12.2     *     *
   8  Tower         11.4     *     *
   9  Urban         12.5     *     *
  10  Residence     11.2  11.3     *
 :
READ      [PRINT=data,errors] nr2, tvariety2, loc[4...7]
   1  Ritmo         11.3  10.8  10.5  10.7
   4  Bercy         11.9  11.5  11.1     *
   5  Versailles    11.3  10.6  10.6     *
  10  Residence     11.8  11.5  11.2  11.3
  12  Riant            *  11.4     *  11.2
  13  Semper        11.3  11.8  12.2  10.9
  14  'PBIS 95/91'     *  11.3     *  11.5
  15  'Ceb 9607'       *  11.4     *  11.2
 :
WEAVEVECTORS FIRSTSET=!p(nr1, tvariety1, loc[1...3]) ; \
             SECONDSET=!p(nr2, tvariety2, loc[4...7]) ; \
             COMBINATION=!p(nr, variety1, new[1...3], variety2, new[4...7])
PRINT        nr, variety1, new[1...3], variety2, new[4...7] ; \
             FIELD=5,13,3(7),13,4(7) ; DECIMALS=0,8(1)
```