GAUSSX™

ECONOTRON SOFTWARE, INC.

Version 10.1

Jon Breslaw

January, 2011

Support:

Econotron Software
447 Grosvenor Ave.
Westmount, P.Q. Canada
H3Y-2S5

Tel: (514) 939-3092
Fax: (514) 938-4994
Eml: support@econotron.com
Web: http://www.econotron.com

Contents

A Appendices

B Index

Concept 1

1.1 Overview

There now exist a substantial number of statistical packages that are capable of being run both on Windows and UNIX platforms. An ideal package is fast, flexible and fun. Traditional packages, such as SPSS, SAS etc, are relatively easy to use, since they are command driven. However, they are not particularly flexible, in the sense that if a particular procedure is not implemented by the package (eg. Tobit) then it is not usually possible for the user to add such a procedure. At the other extreme, one can write statistical procedures for oneself in C or FORTRAN. This provides a great deal of flexibility, and can be relatively fast, since one can dispense with overhead. On the other hand, it is definitely not fun.

GAUSS is a programming language which starts to come close to the ideal. It provides the power, speed, and flexibility of a compiled language, such as C, while being easy to learn and use. It is also flexible, in that the user can program in GAUSS to implement any statistical procedure. Thus GAUSS is a distinct improvement over other existing statistical packages.

However, for many potential GAUSS users, the overhead involved in setting up the various modules, and learning the GAUSS programming language is significant, especially if their previous programming skills heave been restricted to high level languages. And even for experienced GAUSS users,

there is a considerable cost in putting together code each time in order to undertake a particular task, while also having to concern oneself with file manipulation, etc. Put another way, it is still considerably easier to program in a higher level language than in a lower level language.

GAUSSX rectifies this situation, by acting as a shell for running GAUSS. It is very easy to use, since no GAUSS programming nor file management knowledge is required. Indeed, this ease of operation means that GAUSSX is useful not only to users who are new to GAUSS , but also to experienced GAUSS programmers. Under UNIX or MAC, GAUSSX runs as an application, while the Windows version provides a Project Options screen with additional menu items and tools.

While new users can run GAUSSX without any knowledge of GAUSS, all of the data transformation commands that are available to GAUSS are also available to GAUSSX, and thus the new user is eased into GAUSS as his/her needs grow. For experienced GAUSS users, the ability to modify GAUSSX, or to add modules to GAUSSX, makes GAUSSX the preferred environment for running GAUSS. In addition, the GAUSS Application Modules can also be run from GAUSSX. For pedagogical use, GAUSSX can be used to teach econometrics, without the problem of the student spending all of his or her time learning programming, and not econometrics.

Most of the GAUSSX commands are similar to those found in TSP. GAUSSX supports data input either coded in the command file (LOAD), or through reading external files, in either ASCII, spread sheet, or GAUSS format. Data transformation (GENR) is limited only by the GAUSS command set—almost anything. COVA generates descriptive statistics, SVD, and correlograms, while TABULATE provides statistics in a cross-tab format. PRINT, PLOT, and GRAPH commands are supported. Sample selection (SMPL) is supported either directly, or logically, and missing values are handled automatically. A full range of linear single and multiple estimation methods are available, including ARIMA, EXSMOOTH, KALMAN, PANEL, QR and ROBUST. Diagnostics are provided for most single equation methods. Non linear estimation methods include FIML, GMM, ML, NLS, non linear 2SLS and 3SLS, and transfer functions. Linear and non-linear parameter constraints are available for all non-linear estimation routines. A large range of processes are provided for ML estimation, including univariate and multivariate garch, MNL, MNP, neural networks, non-parametric, duration models, Kalman filter, ARFIMA, and stochastic volatility process.

Forecasting methods include both static and dynamic forecast (FORCST), and dynamic solutions for systems of non-linear equations (SOLVE). About 20 specification tests are available using the TEST command including specification, cointegration and decomposition tests, as well as 12 non-parametric tests. Monte Carlo simulation is supported using MCS, and Bayesian estimation using MCMC. Since GAUSS and GAUSSX code can be intertwined, comments, logical goto, and looping, as well as any legal GAUSS commands are permitted.

While GAUSSX takes care of the econometrics, it also provides a number of utilities that facilitate housekeeping. One such utility is project management control. If one is involved in a number of different tasks (or projects), it becomes tedious changing the path and file names each time one changes projects. Each project has a unique name, and specifies the files and paths associated with that project. This way, you don't have to bother with remembering which files are associated with each project. Project management can be used for both GAUSSX and GAUSS projects.

GAUSSX is flexible – you can run it the way you want to. The screen can be toggled on or off, so can the printer. The output file can include the program listing first, if you wish. And of course context sensitive hypertext help is provided in Windows. Graphic support uses the Publication Quality Graphics package, or, if installed, the GAUSSPlot package. Data conversion between foreign file types and GAUSS format is also provided. Access to econometric links on the web - both data and code - is facilitated directly from GAUSS. GAUSSX also provides links to other installed Windows programs, include symbolic processing using Maple or Mathematica.

The remainder of this manual describes how to install and run GAUSSX. If you are a new user, you should read Chapter 2 – installation, and Chapter 3 – running GAUSSX. Then, after GAUSSX is installed, you should get a feel for the program by going through the command files provided - these are on the gauss\prg folder. The first is tutor.prg, which explains how a typical GAUSSX command file is written, and how to use the context sensitive help. Have a look at this file by using the EDIT command. Then try executing it using the RUN command, and then review the output using the VIEW command. There are a number of other command files available (test01.prg - test57.prg); these files will give you an idea as to how the GAUSSX command files are written, and can act as templates for your own work. Details for each command are given in the reference chapter. Finally, to get an idea of what is new, look at the file readme2.txt, on the \gauss\gsx\doc folder.

2 Installation and Configuration

2.1 Installing GAUSSX

The installation routines decompress and copy files and libraries from the compressed file to your GAUSS folder, and then compiles the GAUSSX routines. Before starting however, read this chapter.

As for all software, please observe the normal rules for copyright material.

It is assumed that you have GAUSS already installed in a directory \gauss, and that it is working properly. In addition, you should have set the defaults for your machine for GAUSS in gauss.cfg,

GAUSSX runs under GAUSS, and consequentially will run in any environment that supports GAUSS. The appropriate version of GAUSSX corresponds to the version of GAUSS that you have installed.

2.1.1 GAUSSX for Windows

GAUSSX for Windows runs as a 32 bit Windows application when run under a 32 bit version of GAUSS , and as a 64 bit version when run under a 64 bit version of GAUSS . GAUSS for Windows

6.0 and higher is required. Windows 2000, XP, Vistaand Windows 7 are supported. Automatic differentiation requires Maple 9 or higher.

1. GAUSSX for Windows is distributed as a zip file, typically gaussx_vsn.zip. Unzip the file in a temporary folder, and then execute the file setup.exe. This will do the necessary installation, extraction, and create a program group. You will be prompted for the directory in which GAUSS is installed, and for your GAUSSX license ID which was provided when you purchased GAUSSX. The GAUSSX license ID for the student version is 0. The GAUSSX program files are compiled by GAUSS during the installation process.

2. From Windows, either run GAUSSX from the GAUSSX program group, or execute GAUSS in the normal way, and at the GAUSS prompt type:

   run gaussx ←┘

3. If you have problems installing GAUSSX for Windows, check the Appendix: Trouble Shooting.

2.1.2 GAUSSX for UNIX and MAC

GAUSSX for UNIX is written entirely in GAUSS, and thus is machine independent. It can run either in terminal or X-window mode. GAUSS for UNIX 6.0 or higher is required.

1. Insert the GAUSSX CD, and at the UNIX prompt type:

   cd gauss_directory_name
   tar –xpvf cd_device_name
   tgauss –b gsx/gaussx.cpl

   gauss_directory_name is the system name of the GAUSS directory, and cd_device_name is the system name for the cd drive. The files will then be extracted and copied to the GAUSS directory, and then compiled under GAUSS.

2. Execute GAUSS in the normal way, and at the GAUSS prompt type:

   run gaussx ←┘

3. The first time you run GAUSSX, it searches for the configuration file gaussx.cfg on the /gauss/gsx path; if it does not find it, it creates it. This is a text file that you can edit with a text editor. See the sections on Network Support and Configuration below.

4. If you have problems installing GAUSSX for UNIX, check the Appendix: Trouble Shooting.

2.1.3 Installing GAUSSX manually

Users with non-standard configurations may undertake parts of the installation process manually. In particular, if the folder \gauss\cp is empty, then compilation did not occur during the installation process.

To compile the GAUSSX files, enter GAUSS, and at the GAUSS prompt type:

run gsx\gaussx.cpl ←┘

This will read the GAUSSX source programs, compile them, and save them on \gauss\cp. It will also create the GAUSSX compiled file, gaussxcp.gcg.

2.2 Configuring GAUSSX for Windows

2.2.1 GAUSSX mode

GAUSSX can be used to run both GAUSS and GAUSSX command files; the default is GAUSSX . The mode is shown in the Project Options screen under File Type - it displays either GAUSS or GAUSSX. The state can be toggled using the Option\Parse menu item.

2.2.2 Precision

GAUSSX operates under a default of double precision - all data are written using an 8 character byte. This allows for 15-16 digit precision. If you need only single precision, you can use the statement:

option single;

within the GAUSSX command file to allow for a 4 character byte, permitting 6-7 digit precision. The cost of double precision is a doubling in file length for the scratch files. If you use ATOG under GAUSSX, it will operate at the same precision as currently specified. Stand alone ATOG has a default of single precision.

2.2.3 Projects

You can import an existing project using the File / Import Project File menu item from the GAUSSX Project Options screen.

2.2.4 Configuration File

The GAUSSX configuration file is gaussx.cfg on the gauss\gsx subdirectory. Windows settings are specified under the [Windows Configuration] section.

- gaussx_option determines whether the Project Options menu is displayed at start up. Default is on.
- Excel_process determines whether spreadsheet files are read and written using Excel (which must be installed), or using data exchange. For importing Excel 97 and higher, Excel_process is always utilized. Default is on.
- graphic_support determines whether Publication Quality Graphics (PQG) or GAUSSPlot (GPLOT) is used for rendering graphs. Default is PQG.

2.2.5 Performance

- Set cache_size in the GAUSS configuration file (gauss.cfg) to the correct value.
- From the GAUSS menu item Configure\Editor, set the font to Fixedsys. The other fonts result in considerably slower screen display.

2.2.6 Network support

Each user maintains a project file windesk.prj which resides on gauss\gsx. Normally, the files in this subdirectory reside on the system disk. GAUSSX will look for an environment variable called GAUSSXPATH. If this is not found, GAUSSX assumes that the configuration files can be read and written to \gauss\gsx. If the environment variable is found, GAUSSX will read and write these files on the path specified by the environment variable.

Configuration information for the Project Options Screen and for GAUSS are stored for each user (hkey_current_user) in the Windows registry. Registration information for GAUSSX is stored in (hkey_local_machine) in the Windows registry; this registry item (software\econotron\Gaussx) may need to be exported to the client, depending on the operating system.

2.2.7 Student version

The student version of GAUSSX will only run under GAUSS-Light, which is the student version of GAUSS. All limitations that relate to GAUSS-Light obviously thus carry over the student version of GAUSSX. Installation for the student version of GAUSSX is identical to the procedures described above. Please note that technical support is available through your professor, and that no direct technical support for the student version is provided.

2.3 Configuring GAUSSX for UNIX

2.3.1 UNIX configuration file

The GAUSSX for UNIX system configuration file is gaussx.cfg on the /gauss/gsx subdirectory. It will be copied to the path specified by the environment variable GAUSS_CFG. This should be edited before first running GAUSSX - make sure that all the path names are valid – UNIX is case sensitive. The configuration options are:

gaussx_editor	[pico]	The editor used by GAUSSX
gaussx_viewer	[pico]	The viewer used by GAUSSX
display_mode	[default]/term/X-window	Terminal type
screen_clear	[fast]/slow	Terminal mode screen clear
printdos	enable/[disable]	GAUSS printdos command
graphic_support	gplot/[pqg]	Graphic package

2.3.2 Network support

Each user maintains a configuration file gaussx.cfg which resides on /gauss/gsx. GAUSSX will look for an environment variable called GAUSS_CFG, typically specified in the user's PROFILE. If this is not found, GAUSSX assumes that the configuration files can be read and written to /gauss/gsx. If the environment variable is found, GAUSSX will read and write this file on the path specified by the environment variable. This file should be edited to ensure that the path names are valid.

2.3.3 Porting PC files

Each line of a PC text file finishes with a LF/CR, while in UNIX each line finishes with LF. Thus GAUSSX (and GAUSS) programs written for the PC need to be converted. In addition, GAUSSX and GAUSS data files (dat) created on a PC may not work on a UNIX machine. An easy method of porting these files is to first archive all the relevant files using pkzip. Then transfer the archive over the net to the UNIX machine in binary mode. Finally unzip the archive using the –a option. GAUSS and GAUSSX data files can then be translated to GAUSS's UNIX format using Aptech Systems' transdat program.

Running GAUSSX under Windows 3

GAUSSX is executed from Windows by clicking the GAUSSX icon on the desktop, or by executing run gaussx; from the GAUSS prompt.

GAUSSX for Windows runs under GAUSS, using an additional toolbar to provide access to the GAUSSX controls. Some of these tasks can also be carried out from the Project Options screen, which, under the default GAUSSX configuration setting, is the first screen that is displayed on launching GAUSSX for Windows . The Project Options screen can also be displayed by clicking the Options button in the GAUSSX toolbar.

3.1 Quick Start

The default project has a command file called tutor.prg. Make sure that paths shown are valid. Click EDIT; the file is displayed in the GsxEdit. Now run this file by clicking the RUN button. You can view the output file after execution by clicking the VIEW button.

3.2 Project Options

3.2.1 Overview

GAUSSX provides project management control. If one is involved in a number of different tasks (or projects), it becomes tedious changing the path and file names each time one changes projects. GAUSSX for Windows permits up to 100 projects to be maintained. Each project, which has a name, mode, and description, is linked to the files and paths associated with that project. An existing project can be opened from the toolbar button, and projects can be created, opened, deleted, renamed, imported and exported using the menu item.

The Project Options screen displays the files and paths that are associated with the current project. The project name is shown in the top right panel, and the files and paths are shown in the file display area. The project mode - whether the command file is written in GAUSS or GAUSSX - is indicated by the `File Type` box.

3.2.2 Files

Files and/or paths can be selected by cursor control and ¡enter¿, by clicking with the mouse, or by typing the highlighted letter.

Command File The `COMMAND FILE` tells GAUSSX the path and name for the command file; this is stored in the global variable `_INFILE`; It is often a good idea to keep the command files and data for a particular project in the same subdirectory. Typically, programs are stored on the subdirectory `\gauss\prg`. The command file is where the GAUSSX program is written. The GAUSSX command language is particularly simple - (it is based on TSP) - and full documentation is provided in this manual. The test programs `test01.prg` - `test57.prg` provide some examples of GAUSSX command files. Context sensitive help for GAUSSX syntax is available while editing the command file. See the program tutor.prg for a tutorial.

Output File The `OUTPUT FILE` path tells GAUSSX the path and name for the output file; this is stored in the global variable `_OUTFILE`. GAUSS writes the results of the statistical analysis being undertaken to this file. It is an 80 column ASCII file, and can be imported as text into any word processor. Generally, it is a good idea to name the output from the first run output1.doc, from the second output2.doc, etc.

Data Path The DATA path sets the global variable _PATHD. It tells GAUSSX the path for ASCII, GAUSS, or GAUSSX data that is to be read prior to a GAUSSX analysis. Files saved under GAUSSX will also use this path.

Work Path The WORK path sets the global variable _PATHW. The WORK path tells GAUSSX the path to use for the temporary files that are created when data is loaded or transformed. Thus typically GENR and LOAD will both use the WORK files. These files contain the entire sample as specified in the CREATE statement. The parsed input file - gxfile.prg - is also written on this path. In the default, this path is defined by the TMP environment variable.

Sample Path The SAMPLE path sets the global variable_PATHS. It is used by GAUSSX as the path for the temporary file that is used by all commands that require iteration - such as FIML and NLS. As its name suggests, only the data pertaining to the current sample is maintained on this file. In the default, this path is defined by the TMP environment variable.

3.2.3 Options

The Options menu permits a number of GAUSSX options to be set from the Project Options screen. Each option is either ON or OFF; the ON status is shown by a check-mark. An option can be toggled by highlighting the option using either the up/down cursor keys and typing ¡enter¿, or alternatively by typing the respective hot-key shown in the pop-up menu.

Lines If LINES is set to ON, the command #LINESON is placed at the beginning of the parsed command file. Should an error occur, GAUSS will report the line at which the error occurred. LINES ON is the default. If LINES is set to OFF, the command #LINESOFF is placed at the beginning of the parsed command file. This makes for slightly faster execution.

Screen If SCREEN is set to ON, all output is sent to the screen as well as to the output file — this is the default. Setting SCREEN to OFF will speed things up if there is a lot of output — make sure that you do not use a "wait;" command in such a situation. This option can be changed dynamically within the command file.

Print If PRINT is set to ON, all output is sent to device LPT1, and the output file is not written. The default for PRINT is OFF. This option can be changed dynamically within the command file.

Parse Each project has an associated mode, which defines whether the command file is written in GAUSS or GAUSSX . This mode is indicated by the icon on the top RHS of the screen.

A GAUSSX command file is first parsed before being executed by GAUSS, while a GAUSS command file needs no parsing. The mode can be changed by using the PARSE option. A file which includes the statement `library gaussx` will always be taken as a GAUSS command file

Compressed The default output width of GAUSS output is 80 columns. By setting COMPRESSED to ON, 132 column output can be generated. If you plan to send this output to the printer, then a line print mode (16.6 cpi) will work well. You must set the escape codes manually, or set the printer into line mode from the control panel.

3.2.4 Commands

The commands are shown on the toolbar - they are also available from the menu bar.

3.2.5 Configuration

This menu item permits the user to specify external applications used by GAUSSX.

Editor The default editor is GsxEdit, and this editor provides context sensitive help on any GAUSSX reserved word on typing "F1" when the caret is placed on the word. GAUSSX can use any user specified editor, including GAUSS.

Viewer The default viewer is Notepad – any external program capable of viewing files, including GAUSS, can be specified.

Maple This configures GAUSSX for the command line version of Maple.

Mathematica This configures GAUSSX for the command line version of Mathematica.

3.2.6 Navigation

To navigate within the file display and execution control areas use the cursor or the ”TAB” and ”Shift-TAB” keys to move to the required entry. The entry can be executed by typing ¡enter¿ when it is highlighted, or by typing the first letter of the command. Thus to `VIEW` an output file, type the letter ”v”.

KEY	FUNCTION
Tab/shift Tab	Move one field up or down
Highlight letter	Execute respective field
Alt-letter	Highlight Pull-down menu

3.3 GAUSSX Commands

GAUSSX operates within GAUSS for Windows by adding a toolbar attached to the top right corner of the screen. When GAUSSX is first run, the GAUSSX Project Options screen is displayed. Subsequently, the toolbar containing five buttons is displayed when required.

Editing and running GAUSSX command files is identical to editing and running standard GAUSS files, with the following comments:

- Running a GAUSSX command file requires the File Mode to display GAUSSX in the Project Options dialog.
- To edit the command file currently specified in the Project Options Screen, click the GAUSSX `Edit` button. After you have finished editing the file, save it before running the GAUSSX job.
- To run the current command file, click the GAUSSX `Run` button. The GAUSSX toolbar will disappear, since it is not needed. It will reappear at the end of the GAUSSX job.
- To view the output file at the end of a GAUSSX job, click the GAUSSX `View` button.
- If a GAUSS error occurs during the execution of a GAUSSX command file, enter the command `gaussx;` from the GAUSS prompt. This will display the section of the parsed file where the error occurred.

- To return to the Project Options screen, click the GAUSSX `Option` button.
- To return to GAUSS, click the GAUSSX `Exit` button.

3.4 GAUSSX Tools

These tools are accessed from the GAUSSX `Tools` menu.

Execute Maple This facility permits the use of symbolic algebra in GAUSS. GAUSS is a numerical processing language, as opposed to a symbolic language. Thus GAUSS cannot evaluate indefinite integrals, or analytic gradients. The `MAPLE` or `MATHEMATICA` commands permits symbolic operations to be embedded within a GAUSS or GAUSSX command file. These operations include symbolic differentiation and integration, exact linear algebra, and the symbolic solutions to algebraic equations. In addition, a large class of functions that are available in Maple now become accessible to GAUSS.

In this section, we describe how to evaluate symbolic operations using Maple; the same operations also apply in using Mathematica.

Select the Tools\Maple menu item from the GAUSSX Project Options screen. In the (top) input text box, enter the Maple code, and then click the `Submit` button.

The result is displayed in the output text box, showing both the entire Maple session, as well as the equivalent code as a set of optimized Fortran expressions. This code has a syntax that is very close to GAUSS syntax; however some editing is necessary – one needs to convert the Fortran exponent "**" to GAUSS "$\hat{}$" , and ";" must be added to the end of each line. In addition, some Fortran functions have different names than their GAUSS equivalent. After editing, the set of expressions can then be pasted back to the command file. Examples are given in test23.prg.

The command line version of Maple V, rev 4 or higher must already have been configured in the `Configure\Maple` menu item. Maple is available from Maplesoft, Waterloo Maple Inc., Ontario, Canada.

Execute Mathematica See the discussion above for "Executing Maple".

The command line version of Mathematica 3 or higher must already have been configured in the `Configure\Mathematica` menu item. See test27.prg for examples. Mathematica is available from Wolfram Research, Champaign, IL, USA.

Internet Resources This option provides access to the internet from within GAUSS, and provides links to econometric data and GAUSS code. The source file is econolink.htm, which is located on the gauss\gsx folder.

3.5 Batch mode

Batch mode Windows processing can be initiated from the command line using the command:

tgauss -b gaussxb ←┘

In batch mode, the Project Options menu is not displayed. Rather, the files and path are read from the [Project] section of the GAUSSX configuration file. The system exits at the end of the run.

In this mode, all requests for keystrokes are disabled, and no output is displayed. However the output will still be written to the specified output file.

Running GAUSSX under UNIX 4

GAUSSX is executed under UNIX by the command

tgauss gaussx ↵

or from GAUSS for UNIX by typing at the GAUSS prompt:

run gaussx ↵

If there is a GAUSS error, control is returned to GAUSSX by typing:

gaussx ↵

or

run gaussx ↵

4.1 UNIX menu

The UNIX version of GAUSSX was designed to run in terminal mode; consequently, control occurs through a GAUSS menu. As in the Windows version, the command files and data paths must be

specified - see section 3.2.2. The menu supports the commands `EDIT, RUN, VIEW, QUIT` and `EXIT`. Menu choices are made by typing the first letter of the respective command. The default editor and viewer is `vi`. Typing "h" provides a description for each menu option.

4.2 Quick Start

The default configuration file is displayed on running GAUSSX. Make sure that paths shown are valid. First view the default command file (tutor.prg) using the edit facility by entering:

E ↵

The file is displayed in the default editor. On exit from the editor, the configuration file is redisplayed. Now run this file by entering:

R ↵

and then view the output file by entering:

E ↵

4.3 Batch mode

Batch mode UNIX processing can be initiated from the command line using the command:

tgauss -b gaussxb ↵

This will run the current GAUSSX configuration file, and then exit. In this mode, all requests for keystrokes are disabled, and no output is displayed. However the output will still be written to the specified output file.

GAUSSX Commands - Syntax and Summary 5

5.1 Syntax

Each GAUSSX command has a standard syntax, as follows:

COMMAND (dopt) vlist ;
OPTION = opt1;
OPTION = opt2;

where:

COMMAND	is a GAUSSX command
dopt	is an optional set of display options
vlist	is a list of vectors
OPTION	is a GAUSSX subcommand
opt's	are GAUSSX options.

5.2 Variable names

Variable names must be alpha-numeric, and not more than 8 characters in length. However, when using lags, the entire string (eg GNP(-1)) must be less than 8 characters. The first character must be "_" or alpha. GAUSSX is not case sensitive.

Reserved variable names are:

C	—	A vector of unity.
_ID	—	A sequential vector, depending on the frequency.
_SAMPLE	—	A vector of unity if observation is in current sample, else zero.
NOBS	—	The number of observations in the current sample.
N	—	The number of observations in the current dataloop.

Other reserved variable names are described under the heading "Outputs" for each GAUSSX command in the reference section.

5.3 Summary

The commands are arranged alphabetically. For easy reference, a summary of commands arranged by type is given below. Note that GAUSSX will operate much more efficiently if commands of similar type are grouped together—for example, all the data declaration files are grouped, followed by a group of estimation commands. This reduces the number of times the GAUSSX sample file needs to be created, and also reduces the amount of code swapping.

5.3.1 Data creation and handling

CREATE — Creates a new workfile
DENOISE — Noise filter using wavelets
DGP — Data generation process
DIVISIA — Create chained price index
DROP — Removes variables from workfile
DUMMY — Create dummy variables
EXPAND — Expands a matrix in quad and cross terms
FETCH — Fetch data for global operations
DURATION — Duration model measures
FEVAL — FRML evaluation
FILTER — Data filter
FORCST — Forecast or create variables
GENR — Generates a new series from a formula
GETF — Loads a matrix from disk
KEEP — Retains variables in workfile
LOAD — Loads data into GAUSSX
NORMAL — Transforms vector to a normal variate
OPEN — Reads a data file into GAUSSX
PDL — Defines PDL variables
PRIN — Principal components
PUTF — Saves a matrix to disk
RENAME — Renames a variable in a workfile
SAMA — Seasonal adjustment
SAVE — Saves the current work file onto disk
SOLVE — Solve a system of equations
STORE — Store global data in GAUSSX workspace
SURVIVAL — Survival model measures

5.3.2 Descriptive

ANOVA	— Analysis of variance
CATALOG	— Descriptive comments
CLUSTER	— Cluster groups and dendrogram
CORDIM	— Correlation dimension
CORR	— Correlation measures
COVA	— Correlation matrix & descriptive statistics
CROSSTAB	— Cross-tabulation of data
FREQ	— Frequency distributions
GRAPH	— Graph one variable against another
LYAPUNOV	— Lyapunov exponent
PLOT	— Plot series against time
PRINT	— Print vectors
SVD	— Singular value decomposition
TABULATE	— Descriptive statistics in a hierarchical table
TEST	— Parametric and non-parametric test statistics

5.3.3 Formula definition

ANALYZ	— Parameter generation
CONST	— Constant definition
FRML	— Formula & macro definition
PARAM	— Parameter definition

5.3.4 Estimation methods

AR	— Autoregressive errors
ARCH	— Autoregressive conditional heteroscedastic errors
ARIMA	— Autoregressive integrated moving average
EXSMOOTH	— Exponential smoothing
FIML	— Full information maximum likelihood
GMM	— General method of moments
HECKIT	— Heckman sample selection model
KALMAN	— Kalman filter
LP	— Linear programming
MCMC	— Markov chain Monte Carlo
MCS	— Monte Carlo simulation
ML	— Maximum likelihood
NLS	— Nonlinear least squares
NPR	— Nonparametric regression
OLS	— Ordinary least squares
PANEL	— Panel data regression
PLS	— Partial least squares
POISSON	— Poisson regression
QR	— Quantal response (logit, probit, ordered)
ROBUST	— Robust estimation
RSM	— Response surface methodology
STEPWISE	— Stepwise regression
SURE	— Seemingly unrelated regression estimation
VAR	— Vector autoregressive
2SLS	— Two stage least squares
3SLS	— Three stage least squares

5.3.5 Processes

AGARCH	— Asymmetric GARCH process
ANN	— Artificial neural network
ARCH	— ARCH process
ARFIMA	— ARFIMA process
ARIMA	— ARIMA process
ARMA	— ARMA process
BETA_D	— Beta (distribution) process
COX	— Cox proportional hazards model
DBDC	— Double-bounded dichotomous choice process
EGARCH	— Exponential GARCH process
EXPON	— Exponential process
FIGARCH	— Fractionally integrated GARCH process
FMNP	— Feasible multinomial probit
FPF	— Frontier production function
GAMMA_D	— Gamma (distribution) process
GARCH	— GARCH process
GOMPERTZ	— Gompertz process
GUMBEL	— Gumbel (largest extreme value) process
IGARCH	— Integrated GARCH process
INVGAUSS	— Inverse Gaussian process
KALMAN	— Kalman filter
LOGISTIC	— Logistic process
LOGLOG	— Loglogistic process
LOGIT	— Binomial logit process
LOGNORM	— Lognormal process
MGARCH	— Multivariate GARCH process
MNL	— Multinomial logit
MNP	— Multinomial probit
MSM	— Markov switching models
MVN	— Multivariate normal process
NEGBIN	— Negative binomial process
NORMAL	— Normal process
NPE	— Non parametric estimate
ORDLGT	— Ordered logit process
ORDPRBT	— Ordered probit process

PARETO — Pareto process
PEARSON — Pearson process
PGARCH — Power GARCH process
POISSON — Poisson process
PROBIT — Binomial multivariate probit process
SEV — Smallest extreme value process
SV — Stochastic volatility process
TGARCH — Truncated GARCH process
TOBIT — Tobit process
VARMA — Vector autoregressive moving average process
WEIBULL — Weibull process
WHITTLE — Local Whittle process

5.3.6 Bitwise Commands

IAND — Bitwise and
IEQV — Bitwise eqv
IOR — Bitwise or
INOT — Bitwise complement
ISHFT — Bitwise shift
IXOR — Bitwise xor
RADIX — Convert decimal to base
RADIXI — Convert base to decimal

5.3.7 Statistical Commands

CDF	— Cumulative density function
CDFI	— Inverse cumulative density function
CDFMVN	— Cumulative density multivariate normal
COPULA	— Copula
INVERT	— Inverts a function
LHS	— Latin hypercube sampling
MROOT	— Largest root
MVRND	— Multivariate random sampling
PDF	— Probability density function
PDROOT	— PD Test for smallest root
QDFN	— Multivariate normal rectangular probabilities
RND	— Random sampling from density function
RNDGEN	— Random sampling from any distribution
RNDQRS	— Quasi random number sequences
RNDSMPL	— Random sampling with or without replacement
RNDTN	— Truncated multivariate normal random numbers
STATLIB	— Library of statistical distributions

5.3.8 Finance/Economics Commands

AMORT	— Amortization schedule
FRONTIER	— Efficient frontier
FV	— Future value
GINI	— Gini coefficients
MCALC	— Mortgage calculation
ME	— Maximum Entropy
PV	— Present Value
SPECTRAL	— Power spectrum estimation
WELFARE	— Consumer surplus and deadweight loss

5.3.9 In-Line commands

LAG	— Lag
NMV	— Not missing value
NUMDATE	— Observation number

5.3.10 Support Functions

ACF	— Autocorrelation function
ACV	— Autocovariance function
ARCCOSH	— Inverse cosh function
ARCSINH	— Inverse sinh function
ARCTANH	— Inverse tanh function
CENMEANC	— Censored mean
CENSTDC	— Censored standard deviation
COMBS	— All combinations
DECONV	— Vector deconvolution
INTERP	— Vector interpolation
INTERP2	— Matrix interpolation
ISCHAR	— Test for character vector
ISEMPTY	— Test empty string
LNGAMMA	— Natural log of the gamma function
MPRINT	— Print formatted matrix
PERMS	— All permutations
POLYDIV	— Polynomial division
POLYINV	— Polynomial inverse
SCALZERO	— Test scalar zero
WAITKEY	— Prompt for key input
XGAMMA	— Gamma function
XPAND	— Expand matrix in own and cross powers

5.3.11 Miscellaneous

#LIST	— Enable command file listing
#NOLIST	— Disable command file listing
END	— End of command file
EVAL	— Evaluate string
FMTLIST	— Formats output
GROUP	— Specifies conditional variables
LIST	— Replace variable list
LOADPROC	— Load a previously stored procedure
LOOP	— Loop over block of code for multisector data
NFACTOR	— Memory management
OPTION	— Set GAUSSX options
PAGE	— Page break
SAVEPROC	— Save a symbolic procedure
SMPL	— Specifies the sample
TIMER	— Timer control
TITLE	— Sets a title
?	— Command file comments
@@	— GAUSS commands

5.4 Display Options

The display options specified in ¡dopt¿ consist of the following. Note that each command uses only a subset of these options.

" b" — print brief output
" c" — print correlation matrix
" d" — print descriptive statistics
" e" — print elasticities
" h" — hardcopy option for plots
" i" — print parameters at each iteration
" m" — print marginal effects
" p" — pause after each screen display
" q" — quiet – turn off screen and printed output
" r" — rotate axis
" s" — print diagnostic statistics
" v" — print covariance matrix

5.5 Reference Syntax

The following syntax is used throughout the reference section:

run gaussx;	Command lines.
ARIMA	GAUSSX commands and options.
winsize	Placeholder for user input.
DIRECT	Option values.
test01.prg	File names.
TMP	Other variables.

GAUSSX Reference 6

Purpose

Creates a vector of log likelihoods for an asymmetric GARCH process.

Format

z = AGARCH (*resid, avec, bvec, gvec*);
z = AGARCH_T (*resid, avec, bvec, gvec, dvec*);

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for the ARCH process.
bvec	literal, vector of parameters for the GARCH process.
gvec	literal, vector of γ parameters.
dvec	literal, distributional parameter (ν).

Output

z	vector of log likelihoods.
_ht	vector of conditional variance.

Remarks

The structural coefficients and the coefficients of the AGARCH process are estimated using maximum likelihood. The AGARCH model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ h_t &= \alpha_0 + \sum_{i=1} \alpha_i (|\epsilon_{t-i}| - \gamma_i \epsilon_{t-i})^2 + \sum_{j=1} \beta_j h_{t-j} \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The α are the vectors of the weights for the lagged asymmetric ϵ^2 terms; this is the ARCH process. The β are the weights for the lagged h terms; this is the GARCH process.

avec is a vector of parameters giving the weights for the lagged asymmetric squared residuals. The first element, which is required, gives the constant. *gvec* is a vector of parameters for the asymmetric process - the order of *gvec* should be one less than the order of *avec*. *bvec* is the vector of parameters for the GARCH process. Note the stationarity conditions described under GARCH.

See the "General Notes for GARCH" under GARCH, and the "General Notes for Non-Linear Models" under NLS.

Example

```
OLS y c x1 x2;
sigsq = serˆ2;
PARAM c0 c1 c2;
   VALUE = coeff;
PARAM a0 a1 a2 b1 g1;
   VALUE =  sigsq .1 .1 0 0;
FRML cs1 a0 >= .000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 b1 >= 0;
FRML cs5 a1+a2+b1 <= .999999;
FRML eq1 resid = y - (c0 + c1*x1 + c2*x2);
FRML eq2 lf = agarch(resid,a0|a1|a2,b1,g1|g1);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4 cs5;
```

In this example, a linear AGARCH model is estimated using constrained maximum likelihood, with OLS starting values. The residuals are specified in eq1, and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive. This is a simplified model, in that only one γ parameter is specified – an alternative would be to have a separate γ for each lag.

Source

GARCHX.SRC

See Also

GARCH, EQCON, FRML, ML, NLS

References

Ding, Z., R.F. Engle, and C.W.J. Granger. (1993), "A Long Memory Property of Stock Market Returns and a New Model", *Journal of Empirical Finance*, Vol 1 (1), pp 83-106.

Purpose Calculate the amortization schedule over the life of a loan.

Format {*prin, intrst, balnce* } = AMORT (*p, r, n*);

Input

p	scalar, loan amount.
r	scalar, interest rate per period.
n	scalar, number of periods.

Output

prin	*n*x1 vector of principal payments at each period.
intrst	*n*x1 vector of interest payments at each period.
balnce	*n*x1 vector of remaining balance at each period.

Remarks The AMORT statement returns three vectors, each of length *n* showing the payment of principal and interest for each period of the loan, and the balance outstanding. The interest rate is per period; thus an annual rate of 9% paid monthly for 20 years would have $r = .09/12 = 0.0075$, and $n = 12 * 20 = 240$.

AMORT is pure GAUSS code, and is used independently of GAUSSX.

Example

```
library gaussx ;
p = 1000;
r = .1/12;
n = 12;
{prin,intrst,balnc} = amort(p,r,n);
```

```
prin'=
   79.582554        80.245742        80.914456        81.588743
   82.268650        82.954222        83.645507        84.342553
   85.045407        85.754119        86.468737        87.189310

intrst'=
   8.3333333        7.6701454        7.0014309        6.3271437
   5.6472375        4.9616655        4.2703803        3.5733344
   2.8704798        2.1617680        1.4471504       0.72657758

balnc'=
   920.41745        840.17170        759.25725        677.66850
```

```
595.39985       512.44563       428.80013       344.45757
259.41217       173.65805       87.189310        0.0000
```

This example calculates the interest payment, principal payment and remaining balance for a $1000 loan paid at 10% paid off over one year. If the sample size is set to `n`, the vectors `prin`, `intrst` and `balnc` can be saved in the GAUSSX workspace using the STORE command.

Source FINANCE.SRC

See Also FV, MCALC, PV

Purpose

Estimates the values and covariance matrix for a set of non-linear functions of parameters estimated from the most recent estimation.

Format

ANALYZ (*options*) *elist ;*
METHOD = *method ;*
REPLIC = *nrep ;*
TITLE = *title ;*
VLIST = *vlist ;*

Input

options	optional, print options.
elist	literal, required, equation list.
method	literal, optional, standard error mode. [Delta]
nrep	literal, optional, number of replications. [5000]
title	string, optional, title.
vlist	literal, optional, parameter list.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t–statistics.
VCOV_A	Parameter covariance matrix.

Remarks

The ANALYZ command estimates the parameters on the LHS of each equation specified, as well as the parameter covariance matrix, based on the estimated covariance matrix of the most recent estimation. The coefficient vector from the previous Type I regression must be assigned to parameters in a VLIST statement. The new parameters are then available for use as if they had been created in a PARAM statement.

Two methods are available for estimating the standard error of the estimated parameters:

DELTA This is the standard method for calculating the standard errors, and is the default.

KR The Krinsky-Robb methodology estimates the standard errors using *nrep* simulations of the original parameters, drawn from a multivariate normal distribution with the estimated covariance matrix.

Print options include d – print descriptive statistics, p – pause after each screen display, v – print parameter covariance matrix, and s – compute the Wald statistic for the hypothesis that the set of functions is jointly zero.

Example

```
FRML eq1 y1 = a0 + a1*x + a2*z;
FRML eq2 y2 = b1 + ln(b2*x + a2*w);
FRML eq3 c1 = (b1 + b2 + b3)/sigma;
FRML eq4 c2 = (b1/a1);
FRML eq5 elas = a1*meanc(y)/meanc(x);
NLS eq1 eq2;

1.   ANALYZ (p,d,v,w) eq3 eq4;

2.   ANALYZ (p,s) eq5;

3.   OLS y c x1 x2;
     FRML eqa1 bb1 = b1/b2;
     ANALYZ (p) eqa1;
        METHOD = KR;
        VLIST = b0 b1 b2 ;
```

These examples show typically how ANALYZ is utilized. Two Type II equations are jointly estimated using NLS – in fact this is a non-linear constrained estimation, since the coefficient on w is constrained to be the same as the coefficient on z. Parameters c1 and c2 are created - their values and standard errors are displayed (d), as is their variance-covariance matrix (v), and the Wald statistic (s) is computed.

The second ANALYZ statement shows how an elasticity at the mean could be evaluated, along with its variance. Note that the function must return a scalar - GAUSSX looks after fetching the data.

The third example shows how ANALYZ is used after a Type I estimation, using the Krinsky-Robb methodology. The number and position of parameters in *vlist* relate directly to the coefficient vector of the previous estimation.

See Also CONST , FRML , NLS , PARAM

References Rao C.R. (1973), *Linear Statistical Inference and its Applications*, Wiley, New York.

Krinsky, I., and A. L. Robb. (1986), "On Approximating the Statistical Properties of Elasticities." *Review of Economics and Statistics*, Vol 68, pp. 715-719.

Purpose Returns the fitted values of an Artificial Neural Networks process.

Format
z = ANN (*x, amat, bmat*);
z = ANN (*y~x, amat, ny*);

Input

x	literal, required, matrix of inputs.
amat	literal, required, matrix of parameters for hidden layer.
bmat	literal, required, matrix of parameters for output layer.
y	literal, required, matrix of outputs.
ny	literal, required, number of outputs.

Output *z* Matrix of predicted values or probabilities.

Remarks ANN is used as part of a FRML statement to estimate the hidden and output weights of a neural networks process. This is achieved using either least squares in a regression context, or maximum likelihood in a probability context. An econometric formulation of a feed forward (ie. non-recursive) single hidden layer ANN is:

$$y_h = F\left(\beta_{h0} + \sum_{j=1}^{q} G(\widetilde{x}'\gamma_j)\beta_{hj}\right) \qquad h = 1, \ldots, g$$

where y_h is a gx1 vector of endogenous variables, $\widetilde{x} = (1, x_1, \ldots, x_k)'$ is a kx1 vector of explanatory variables, $\gamma_j = (\gamma_{j0}, \gamma_{j1}, \ldots, \gamma_{jk})'$ is a $k+1$x1 vector of hidden weights, q is the number of hidden units, G is the transformation applied in the hidden layer, $\beta_h = (\beta_{h0}, \beta_{h1}, \ldots, \beta_{hq})'$ is a $q+1$x1 vector of output weights, and F is the transformation applied in the output layer. (Observation subscripts are excluded for clarity). Commonly, $G(.)$ is sigmoid:

$$G(\widetilde{x}'\gamma) = 1/(1 + e^{-\widetilde{x}'\gamma})$$

though any mapping on the $\{0, 1\}$ space will do. If y is continuous, $F(.)$ should be linear, ie. $F(z) = z$, while if y is a limited dependent variable, $F(.)$ should also map to the $\{0, 1\}$ space. In econometric terms, this is a system of g non-linear equations, with some common coefficients (γ) across equations.

The program control options for both ML and for NLS are described in the "General Notes for Non-Linear Models" under NLS. In addition, there are some specific options available under the OPLIST option:

OPLIST = *progopts* ;

where:

progopts literal, optional, options for ANN control.

The program control options are specified in *progopts*. The options available are:

HIDDEN = *function* Specifies the transfer function carried out in the hidden layer. *function* can be chosen from the following: ARCTAN, CDFN, GAUSSIAN, HALFSINE, LINEAR, SIGMOID, STEPFN, and TANH. The default is SIGMOID.

OUTPUT = *function* Specifies the transfer function carried out in the output layer. *function* is defined above, but also includes OLS. The default is SIGMOID under ML, and LINEAR under NLS.

NONE/DENSITY/MAXIMUM Specifies the type of output scaling that is carried out. NONE invokes no scaling - this is the default under NLS. DENSITY scales the output such that the sum for each observation is unity; this is the default under ML. MAXIMUM does not return a matrix, but instead returns a vector containing the index of the category with the maximum value.

AUGMENT/[TRANSFER] Specifies the type of model estimated. The default is the feedforward model consisting of a single hidden and a single output layer, with a transfer function for each. Under the AUGMENT option, the hidden layer consists of the sum of the hidden transfer function and a linear function of the inputs - consequently β will have k extra elements.

PRINT/[NOPRINT] Specifies whether a description of the ANN options actually used should be printed out. This is useful for debugging.

When the output transfer function is linear and the optimization is undertaken using NLS, it is possible to express the output weights (β) in a closed form. This results in a significant reduction in the number of parameters that need to be estimated. The second form of the ANN command is used, and the output transfer function is specified as OLS; the output weights are available as a global called _nnbeta.

Since ANN returns a matrix if there is more than one output unit, the command should normally be placed in a macro definition that will be referenced in an EQSUB – see the example below.

Artificial neural networks may often have difficulty converging. In addition, initial values are important, and must be chosen in the context of the selected transfer functions. Start off with a small number of hidden units, and work up. Note that it is often possible to use random hidden weights, and to let the output weights do most of the work. An example of ANN estimation is given in test20.prg.

Example

1.
```
PARAM amat;
   SYMBOL = a;
   ORDER  = 4 2;
PARAM bmat;
   SYMBOL = b;
   ORDER  = 3 2;
FRML eqw  w   := ann(x1~x2~x3,amat,bmat);
FRML eq1  y1   = submat(w,0,1);
FRML eq2  y2   = submat(w,0,2);
NLS (p,i) eq1 eq2 ;
   EQSUB = eqw;
   OPLIST = print;
FORCST y1hat y2hat;
```

2.
```
PARAM amat;
   SYMBOL = a;
   ORDER  = 3 4;
FRML eq1  y  = ann(y~x1~x2,amat,1);
NLS (p,i) eq1;
   OPLIST = output = ols print;
```

3.
```
PARAM amat;
   SYMBOL = a;
   ORDER  = 3 1;
PARAM bmat;
   SYMBOL = b;
```

```
   ORDER  = 2 4;
FRML eqp  p   := ann(x1~x2,amat,bmat);
FRML eq0  llf = ln( p[.,1].*y1 + p[.,2].*y2
                  + p[.,3].*y3 + p[.,4].*y4);
ML (p,i) eq0 ;
   EQSUB = eqp;
FETCH x1 x2;
prob  = ann(x1~x2,amat,bmat);
STORE p1hat p2hat p3hat p4hat;
   VLIST = prob;
```

The first example shows an ANN estimation of a continuous variable, three (k) input, two (g) output model, with 2 (q) units in the hidden layer. amat is a 4x2 ((k+1)xq) matrix of hidden weights, and bmat is a 3x2 ((q+1)xg) matrix of output weights. These weights (parameters) are estimated in the NLS command – the macro eqw is evaluated before every call to eq1 and eq2. The default for NLS generates a sigmoid transformation at the hidden level, and no transformation nor scaling at the output level. A print option is specified in oplist. The fitted values are created in the subsequent FORCST.

The second example shows the ANN estimation of a continuous variable, two (k) input, one (g) output model, with 4 (q) units in the hidden layer. amat is a 3x4 ((k+1)xq) matrix of hidden weights. The output weights are not specified since the option output = ols is specified.

The third example shows an ANN estimation for the categorical variable case. There are two (k) inputs, one (q) units in the hidden layer, and four (g) outputs. y1, ..., y4 take the value unity if the respective category is selected, else zero. amat is a 3x1 ((k+1)xq) matrix of hidden weights, and bmat is a 2x4 ((q+1)xg) matrix of output weights. These weights (parameters) are estimated in the ML command. The default for ML generates a sigmoid transformation at the hidden and output level, and a scaling such that the sum of the outputs equals unity - thus p is a matrix of probabilities. The fitted values are created as shown.

Source NEURALX.SRC

See Also EQSUB , FRML , ML , NLS , NPE

References

Kuan, C.M., and H. White (1994), "Artificial Neural Networks: An Econometric Perspective", *Econometric Reviews*, Vol. 13 (1), pp. 1-91.

Webb, A.R., and D. Lowe (1988), "A hybrid optimisation strategy for adaptive feed-forward layer networks", RSRE Memorandum 4193, Royal Signals and Radar Establishment, Malvern, UK.

Purpose

Undertakes an analysis of variance for a single variable.

Format

ANOVA (*options*) *elist ;*
MODEL = *model ;*
TITLE = *title ;*
VALUE = *random ;*
VLIST = *covlist ;*

Input

options	optional, print options.
elist	literal, required, variable list or equation name.
covlist	literal, optional, covariate list.
model	literal, optional model matrix. [1]
random	literal, optional, random vector.
title	string, optional, title.

Output

STATS	Tabular output.

Remarks

ANOVA implements an N-way analysis of variance providing an adjusted (Type III) sum of squares for fixed, random or mixed models, which can be balanced or unbalanced. Covariates are permitted, and for random or mixed models, variance components are reported. The model is specified in terms of nested effects and interaction effects.

The variables are entered in *elist* with the dependent variable first, followed by a list of categorical variables. Alternatively, *elist* can consist of an equation name which has been previously specified in a Type I FRML command. Each categorical variable, which consists of consecutive integer values, will be transformed to dummy variables by ANOVA. Covariates can be entered as a list in *covlist*.

A model is specified in terms of the *K* categorical variables and the groups that are formed from these categorical variables. All models consist of the level 1 categorical variables. The model, specified in *model*, consists either of a scalar specifying the maximum order, or a *K* column matrix of zeros and ones, where each row corresponds to a group. Nested components are also specified in the model field, using the 'n' notation to specify which variable is nested. Nesting occurs before interaction.

Consider the following analysis of variance:

```
ANOVA Y A B C;
```

Y is the variable whose sum of squares is to be assigned on the basis of the three categorical variables A, B and C. Various models are shown below:

```
model = 1;              Groups:  A, B, C

model = 2;              Groups:  A, B, C, AB, AC, BC

model = 3;              Groups:  A, B, C, AB, AC, BC, ABC

model = { 1 1 0,        Groups:  A, B, C, AB, AC, ABC
          1 0 1,
          1 1 1};

model = { 1 n 0,        Groups:  A, B(A), C, AC
          1 0 1 };


model = { 1 0 1,        Groups:  A, B(A), C, AC, B(A)C
          1 n 0,
          0 1 1 };
```

Similarly, random (and mixed) models are specified in *random*:

```
value =  { 0 0 0};      A - fixed,  B - fixed,  C - fixed

value =  { 1 0 1};      A - random, B - fixed,  C - random
```

Print options includes d —descriptive statistics, q —quiet - no output, p —pause after each screen display, and s —diagnostic checking.

The variable specified in “Outputs” is returned as a global variable.

Example

```
1.  ANOVA (p,d) score noise subject etime;
        MODEL =  2;

2.  FRML eq1 score noise subject etime;
    ANOVA (p) eq1;
        VALUE =  1 1 1;
        MODEL =  { 1 1 0, 0 1 1, 1 1 1 };

3.  ANOVA (p) eq1;
        VALUE =  0 0 1;
        MODEL =  { 1 n 0, 1 0 1 };
```

In example 1, an ANOVA is undertaken on score using noise, subject and etime as categorical variables. The model includes all terms up to level 2 - that is, linear terms and two way interaction terms. Descriptive statistics (d) are displayed.

Example 2 uses the same variables, but this time expressed in a FRML. This is a random model, since each categorical variable is specified as unity in the *random* field. The model consists of the linear terms (always), the two way interactions noise*subject and subject*etime are the three way interaction noise*subject*etime.

Example 3 is a mixed model - etime is random, while the other categorical variables are fixed. This is a nested model - subject is nested within noise - and also includes an interaction term - noise*etime.

References

Macnaughton, D.B, *Computing numerator sum of squares in unbalanced analysis of variance*, http://www.matstat.com/ss/pr0139.sas

Milliken, G.A.and D.E. Johnson (1984), *Analysis of Messy Data*, Van Nostrand Reinhold Co., New York.

Montgomery, D.C. (1991). *Design and Analysis of Experiments*, 3rd ed., J. Wiley and Sons Inc.

Searle, S.R. (1987), *Linear Models for Unbalanced Data*, Wiley, New York.

See Also TABULATE

Purpose

Estimates the coefficients of linear models with serially correlated errors.

Format

AR (*options*) *vlist ;*
MAXIT = *maxit ;*
METHOD = *meth ;*
ORDER = *lags ;*
PDL = *pdllist ;*
TITLE = *title ;*
TOL = *tolerance ;*
VALUE = *values ;*
WEIGHT = *wtname ;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
maxit	numeric, optional, maximum number of iterations (20).
meth	literal, optional, stepsize method (CORC).
lags	literal, optional, AR process (1).
pdllist	literal, optional, options for PDL.
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
values	numeric, optional, starting value of coefficients.
wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t–statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
LLF	Log likelihood.
RSQ	R-squared.
RBARSQ	RBAR-squared.

VCOV Parameter covariance matrix.

Remarks The AR command estimates the parameters of a linear autoregressive model using an iterative procedure. The equation is specified in the same manner as in OLS. Since zero restrictions are permitted, the AR process must be fully specified - see the examples.

Three algorithms are available;

- CORC The Cochrane-Orcutt iterative method. Estimates of rho are derived at each iteration, and the rho-transformed variables are then used for the next iteration. The first observation is dropped.
- GN The Gauss-Newton algorithm. This algorithm generally does better than CORC when there are lagged dependent variables. The first observation is dropped.
- PW The Prais-Winsten algorithm. This algorithm is the same as CORC, but the first observation is not dropped. The PW transformation for the first observation is $\sqrt{(1-\rho^2)}(y_1 - \beta' x_1)$.

The AR process requires that the data must be in core, and uses the current sample, which must be contiguous. GAUSSX automatically drops the first rho cases to allow for the transformed structure.

The coefficient vector (COEFF) consists of the structural coefficients followed by the “rhos”. When used in FORCST, only the structural coefficients are used - thus this is equivalent to an OLS forecast.

The summary statistics are based on the Rho-transformed variables.

See the “General Notes for Linear Models” under OLS, and the example in test07.prg.

Example

```
      FRML eq1 y c x1 x2;
1.    AR (p,d) eq1 ;

2.    AR (p,i) y c x1 x1(-1);
         METHOD = GN;
         ORDER  = 1 2;

3.    AR (v) eq1;
         METHOD = CORC;
         ORDER  = 1 4;
         MAXIT = 40;
```

Example 1 shows the default situation; an AR1 process is modelled, using the Cochrane-Orcutt methodology on eq1; the display pauses (p) after each screen, and descriptive statistics (d) are displayed. In the second example, the equation is specified within the AR command - y is the dependent variable, and c, x1, and x1 lagged once are the independent variables. A second order AR process is modelled - thus there will be two parameters estimated; these are called RHO1 and RHO2. The method used is the Gauss-Newton iterative method. Intermediate result after each iteration are displayed under the (i) option. In example 3, a zero restriction fourth order AR process is estimated - only the first and fourth lags are estimated; the 2nd and 3rd are constrained to zero. MAXIT and TOL act as in NLS. The parameter covariance matrix is displayed under the (v) option.

See Also

FRML , NLS , OLS , PDL , WEIGHT , TITLE

References

Beach, C.M., and J.G. MacKinnon (1978), "A Maximum Likelihood Procedure for Regression with Autocorrelated Errors", *Econometrica*, Vol. 46, pp. 51-58.

Cochrane, D., and G.H. Orcutt (1949), "Application of Least Squares Regression to Relationships Containing Autocorrelated Error Terms", *JASA* Vol. 44, pp. 32-61.

Prais, S., and C. Winsten (1954), "Trend Estimation and Serial Correlation" Discussion Paper 383, Cowles Commission, Chicago.

Purpose

Estimates the coefficients of a linear model with autoregressive conditional heteroscedastic errors.

Format

ARCH (*options*) *vlist ;*
MAXIT = *maxit ;*
MAXSQZ = *iter;*
METHOD = *meth ;*
ORDER = *lags ;*
PDL = *pdllist ;*
TITLE = *title ;*
TOL = *tolerance ;*
VALUE = *values ;*
WEIGHT = *wtname ;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
maxit	numeric, optional, maximum number of iterations (20).
iter	numeric, optional, iterations per equation (3).
lags	literal, optional, AR process (1).
pdllist	literal, optional, options for PDL.
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
values	numeric, optional, starting value of coefficients.
wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t–statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
LLF	Log likelihood.

RSQ	R-squared.
RBARSQ	RBAR-squared.
VCOV	Parameter covariance matrix.

Remarks

The ARCH command estimates the parameters of a linear model in which the errors exhibit non-constant variances conditional on the past variances using an iterative procedure. Generalized ARCH, ARCH–M, and GARCH models are described in GARCH, and multivariate garch models in MGARCH.

The user specifies a FRML in the same manner as OLS, as well as the order of the lag structure of the residuals. GAUSSX uses the current sample to estimate the ARCH process, automatically dropping the first "lagmax" cases to allow for the residual structure. Estimation takes place using the method of scoring. In the default, three iterations are used on the error component, then three on the structural component. This can be changed using the MAXSQZ option. Convergence is not guaranteed. The ALPHAs are not permitted to fall below zero. Starting values for the structural component are estimated using OLS in the default, but can be explicitly given using the VALUE option.

The coefficient vector (COEFF) consists of the structural coefficients followed by the parameters of the error structure. When used in FORCST , only the structural coefficients are used - thus this is equivalent to an OLS forecast.

The summary statistics are based on the variance transformed variables. Thus the residuals should be homoscedastic. Engle's test (Lagrange multiplier) is also shown; this is derived on the original untransformed variables.

The ARCH process requires that the data must be in core, and uses the current sample, which must be contiguous.

See the "General Notes for Linear Models" under OLS. An example of an ARCH model is given in test07.prg. Examples of maximum likelihood methods of estimating linear and non-linear ARCH and GARCH models are given in ARCH and GARCH respectively.

Example

```
FRML eq1 y c x1 x2;

1.   ARCH (p,d) eq1 ;
```

```
2.   ARCH (p,i,s) y c x1 x1(-1);
        MAXSQZ = 2;
        ORDER  = 1 2;

3.   ARCH (v) eq1;
        ORDER  = 1 3;
        MAXIT = 40;
        VALUE = 1 3 .2;
```

In example 1, an ARCH process is modelled based on the default – a one order lag. Thus:

$$\text{var}(e_t) = a_0 + a_1 \, e_{t-1}^2$$

The display pauses (p) after each screen, and descriptive statistics (d) are displayed.

In the second example, the order is now two:

$$\text{var}(e_t) = a_0 + a_1 \, e_{t-1}^2 + a_2 \, e_{t-2}^2$$

Thus three parameters are estimated for the structural form, and three for the error structure; these latter are called ALPHA1, ALPHA2 and ALPHA3. The MAXSQZ subcommand specifies the number of squeezes (iterations) within each loop; the default is three. Intermediate result after each iteration are displayed under the (i) option. The (s) option results in a full set of diagnostic statistics. In example 3, a zero restriction third order ARCH process is estimated - only the first and third lags are estimated; the 2nd is constrained to zero. MAXIT and TOL act as in NLS. The parameter covariance matrix is displayed under the (v) option. Starting values for the structural coefficients is given using the VALUE option.

See Also ARCH , GARCH , OLS , PDL , TITLE , WEIGHT

References Greene, W.H. (1993), *Econometric Analysis* 2nd ed. Macmillan, New York.

Purpose Creates a vector of log likelihoods for an ARCH process.

Format
```
z = ARCH ( resid, avec );
z = ARCH_T ( resid, avec, dvec );
```

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for the ARCH process.
dvec	literal, distributional parameter (ν).

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks The structural coefficients and the coefficients of the ARCH process are estimated using maximum likelihood. The ARCH model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ h_t &= \alpha_0 + \sum_{i=1} \alpha_i \epsilon_{t-i}^2 \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The α are the vectors of the weights for the lagged ϵ^2 terms; this is the ARCH process.

avec is a vector of parameters giving the weights for the lagged squared residuals. The first element, which is required, gives the constant. If only a single parameter is specified, the model is standard OLS. Note the stationarity conditions described under GARCH.

See the "General Notes for GARCH" under GARCH, the "General Notes for Non-Linear Models" under NLS, and the remarks under ARCH. An example is given in test07.prg.

Example

```
OLS y c x1 x2;
sigsq = serˆ2;
PARAM g0 g1 g2;
   VALUE = coeff;
PARAM a0 a1 a2;
   VALUE =  sigsq .1 .1;
FRML cs1 a0 >= .000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 a1+a2 <= .999999;
FRML eq1 resid = y - (g0 + g1*x1 + g2*x2);
FRML eq2 lf = arch(resid,a0|a1|a2);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4;
```

In this example, a linear ARCH model is estimated, using OLS starting values. The residuals are specified in eq1, and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive.

Source

GARCHX.SRC

See Also

ARCH , GARCH , EQCON , FRML , ML , NLS

References

Engle, R.F. (1982), “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of the U.K. Inflation”, *Econometrica*, Vol. 50, pp. 987-1007.

Purpose

Creates a vector of log likelihoods or fitted values for a fractional autoregressive moving average process.

Format

z = ARFIMA (*y, d, phi, theta*);
OPLIST = *progopts ;*

Input

y	literal, *N*x1 vector of time series.
d	scalar, degree of differencing.
phi	literal, *P*x1 AR coefficient vector, or scalar zero.
theta	literal, *Q*x1 MA coefficient vector, or scalar zero.
progopts	literal, optional program options

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The Autoregressive Fractionally Integrated Moving Average (ARFIMA) process permits the estimation of long memory models. The ARFIMA (p, d, q) process is given by:

$$\phi(L)(1-L)^d y_t = \theta(L)\epsilon_t$$

where:

$$\begin{aligned}\phi(L) &= 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p \\ \theta(L) &= 1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q\end{aligned}$$

and where L is the backward shift operator, and d is the fractional degree of differencing.

The coefficients of the ARFIMA process are estimated using either ML or NLS. *y* should be detrended, and have zero mean.

The program control options are specified in *oplist*. The options available are:

CONSTANT/[NOCONST] Specifies whether a constant is to be included. CONSTANT should normally be specified for non-differenced series with

non-zero mean, unless the constant is explicitly specified as a parameter.

Both stationary and invertibility conditions need to be satisfied. GAUSSX provides a routine called MROOT, which returns the value of the largest root which must have a modulus less than unity. In addition, besides the normal AR and MA requirements for stationarity and invertibility, requirements on d include $d > -1$ for invertibility, and $-.5 < d < .5$ for stationarity. Consequently, constrained optimization is usually required.

Estimated values are available after an NLS estimation using the FORCST command. If a range is given, actual values are used up to the first date in the range, and forecast values for the dates up to the second date. Two methods are available - Naive and Best Linear Predictor (BLP). Forecast standard errors are available using METHOD = STDERR.

An example of ARFIMA is given in test43.prg.

See the "General Notes for Non-Linear Models" under NLS.

Example

```
PARAM d phi1 phi2 theta1 theta2;
   VALUE = .5 .5 0 .5 0;
FRML eq1  y =  arfima(y, d, phi1|phi2, theta1|theta2);
FRML ec1 mroot(phi1|phi2) <= .9999;
FRML ec2 mroot(theta1|theta2) <= .9999;
NLS (p,d,i) eq1
   EQCON = ec1 ec2;
   OPLIST = constant;
FORCST yhat;
    METHOD = fit blp;
    RANGE = 1990 1999;
```

This example demonstrates how a $(2, d, 2)$ ARFIMA model is estimated. d is the fractional dimension, and there are two AR coefficients (phi1, phi2) and two MA coefficients (theta1, theta2). The model is estimated using constrained NLS, where the constraints are specified in ec1 and ec2, and where MROOT is a GAUSSX routine for returning the value of the largest root.

Source ARFIMAX.SRC

See Also ARFIMA , ARIMA , MROOT , NLS , VARMA

References

Box, G.E.P., Jenkins, G.M., and Reinsel, G. C. (1994). *Time Series Analysis, Forecasting and Control*, San Francisco: Holden-Day.

Doornik, J.A. and Ooms, M. (1999). "A package for estimating, forecasting and simulating ARFIMA models: Arfima package 1.0 for Ox", *Discussion paper*, Nuffield College, Oxford.

Sowell, F. (1992). "Maximum likelihood estimation of stationary univariate fractionally integrated time series models", *Journal of Econometrics*, Vol. 53, pp. 165-188.

Purpose

Identify, estimate and forecast the autoregressive integrated moving average model.

Format

ARIMA (*options*) *vname ;*
MAXIT = *maxit ;*
METHOD = *meth ;*
NAR = *nar ;*
NDIFF = *ndiff ;*
NMA = *nma ;*
NSAR = *nsar ;*
NSDIFF = *nsdiff ;*
NSMA = *nsma ;*
OPLIST = *progopts ;*
PERIODS = *periods ;*
RANGE = *rangelist ;*
DISPLAY = *screen ;*
TOL = *tolerance ;*
VLIST = *fcstname ;*

Input

options	optional, print options.
vname	literal, required, variable name.
maxit	numeric, optional, maximum number of iterations (20).
meth	literal, optional, algorithm list (GAUSS GAUSS GAUSS).
nar	numeric, optional, number of autoregressive terms (0).
ndiff	numeric, optional, degree of differencing (0).
nma	numeric, optional, number of moving average terms (0).
nsar	numeric, optional, number of seasonal AR terms (0).
nsdiff	numeric, optional, degree of seasonal differencing (0).
nsma	numeric, optional, number of seasonal MA terms (0).
progopts	literal, optional, options for program control.
periods	numeric, optional, number of lags for correlogram (15).
rangelist	numeric, optional, pairs of ranges for forecasting.
screen	literal, optional, screen mode (GRAPH).
tolerance	numeric, optional, param. convergence tolerance (.001).
fcstname	literal, optional, forecast variable name.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
LLF	Log likelihood.
VCOV	Parameter covariance matrix.
_PSTAR	Vector of ϕ^*.
_TSTAR	Vector of θ^*.

Remarks

The ARIMA command undertakes all three parts of the Box-Jenkins process - identification, estimation and forecasting. The ARIMA (p, d, q) process is given by:

The ARIMA (p, q) process is given by:

$$\phi(L)(1 - L)^d y_t = \theta(L)\epsilon_t$$

where:

$$\begin{aligned} \phi(L) &= 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p \\ \theta(L) &= 1 + \theta_1 L - \theta_2 L^2 - \cdots - \theta_q L^q \end{aligned}$$

and where L is the backward shift operator, and d is the degree of differencing.

The ARIMA process requires that *vname* must be in core, and uses the current sample, which must be contiguous. It automatically drops observations for the differencing and AR processes.

Print options include c —print correlogram of the estimated residuals, d —print descriptive statistics, i —print parameters at each iteration, p —pause after each screen display, and q —quiet - no screen or printed output. Additional information is available through the on-line help (Alt-H).

The program control options are specified in *progopts*. The options available are:

[IDENTIFY]/NOIDENT Specifies whether the identification process is to be undertaken.

[ESTIMATE]/NOEST Specifies whether the model is to be estimated.

[FORECAST]/NOFORCST Specifies whether the forecast process is to be undertaken.

CONSTANT/[NOCONST] Specifies whether a constant term is to be included in the model.

[FIT]/RESID Specifies the type of forecast mode.

STATIC/[DYNAMIC] Specifies whether the actual or predicted values of *vname* are used in the forecast process.

PARAM/[NOPARAM] Specifies whether the parameter starting values are to be given in a PARAM or a CONST statement, or whether they are to be evaluated using the Yule-Walker conditions.

[PLOT]/NOPLOT Specifies whether the correlogram and partial autocorrelogram are to be plotted.

See the "General Notes for Non-Linear Models" under NLS. Multi-equation transfer functions can be estimated using the NLS command. Examples of both ARIMA and transfer function estimation are given in test11.prg.

Identification

The identification process is required to determine the degree of differencing necessary to generate a series that is stationary. GAUSSX first provides the correlogram (AC) and partial autocorrelogram (PAC) for the vector *vname* before differencing, and the associated Ljung-Box Q statistics. A plot of the AC and PAC is also provided. This output is then repeated for the series after differencing. Under the default (DISPLAY = GRAPH), the correlograms are displayed using the PQG screen mode. - see 'General Notes for Graphs" in GRAPH. The current sample is used, and GAUSSX drops the first $ndiff + freq * nsdiff$ terms for the differenced series, where *freq* corresponds to the type of data set specified in the CREATE command.

Estimation

The estimation process requires the user to specify the order of the AR ($nar, nsar$) and MA ($nma, nsma$) components, as well as specifying whether

a constant is to be included (OPLIST = CONSTANT). A constant should be specified for non-differenced series with non-zero mean.

GAUSSX will automatically estimate starting values of the parameters of the model, using the Yule-Walker equations. These parameters are called PHI1, PHI2, etc. for the AR parameters, THETA1, THETA2 for the MA parameters, GAMMA1, GAMMA2 for the seasonal AR parameters, and DELTA1, DELTA2 for the seasonal MA parameters. If the option OPLIST = PARAM is specified, starting values for the coefficients must be given by the user in a PARAM or a CONST statement. Thus, if some of the parameters are to be restricted during an ARIMA estimation, they should be specified previously in a CONST statement.

The estimation use the NLS routines, and all the non-linear options are available. The MA component is evaluated recursively each time the residuals are estimated. GAUSSX uses the current sample, and automatically drops the first $ndiff + nar + freq * (nsdiff + nsar)$ observations. Initial values of ϵ are set to the unconditional expected value of zero for the first $nma + nsma * freq$ observations - that is "back-forecasting" is not employed. Parameter values at the end of the estimation are stored both under their individual names, as well as in a global vector called COEFF. A correlogram of the residuals is produced if the c option is specified in *options*. The roots of both the AR and the MA process are reported such that stability and invertibility can be assessed.

Forecasting

The raw coefficients are transformed into the ϕ^* and θ^* vectors, which can be used on the original time series. These are globally available as _PSTAR and _TSTAR. A separate forecast is undertaken for each pair of sample dates specified in *rangelist*, or for the last 15 observations if RANGE is not specified. Under the default (DYNAMIC), the forecasts are based on the actual values of *vname* up to the first element in the pair, and forecast values up to the last element of the pair. Forecasts based on the actual residuals derived during the estimation process can be achieved by using the STATIC option. The vector that is forecast is the fitted value of *vname*, unless OPLIST = RESID is specified, in which case the forecast is the vector of residuals (ϵ). The forecast for the last pair of sample points specified in *rangelist* is stored as a GAUSSX vector under the name given in *fcstname*.

Forecast values for an ARIMA process can also be obtained using the FORCST command immediately following an ARIMA estimation. Both the MODE and the RANGE options must be specified.

Example

```
1.   SMPL 1956 1974;
     ARIMA (p,c,d) y;
        NAR = 1;  NDIFF = 1;  NMA = 2;
        OPLIST = noplot noforcst;

2.   SMPL 19681 19854;
     ARIMA (p,c) q;
        NAR = 2;   NMA = 2;   NSMA = 1;
        OPLIST = const;
        RANGE = 19831 19874  19841 19874;
        VLIST = qfit;

3.   SMPL 1962 1988;
     PARAM phi1 phi2 theta1;
        VALUE = 0 .6 .7;
     CONST phi1;
     ARIMA (p) gnp;
        NAR = 2;   NMA = 1;   NSAR = 1;  NDIFF = 1;
        OPLIST = param noident;
        MAXIT = 40;
```

Example 1 shows how an ARIMA(1,1,2) is undertaken on the vector y. Identification is carried out, followed by estimation, but no forecast is undertaken, nor are the AC and PAC plots produced. A correlogram of the residuals is produced under the c option.

Example 2 shows how a seasonal MA process is modelled. The original series, q is used, since no differencing is specified. Two forecasts are generated, the first from 19831 to 19874, and the second from 19841 to 19874; the latter forecast is stored as the variable qfit, and can be used in subsequent GAUSSX operations.

A restricted model is estimated in Example 3: `phi1` is restricted to zero through the previous CONST statement; while `phi2` and `theta1` take starting values of .6 and .7 respectively. `GAMMA1`, the seasonal MA parameter is not specified, and so takes an initial value of zero. The identification process is bypassed, and after estimation the forecast values for the last 15 observations is displayed, but not saved.

See Also

AR , CONST , EXSMOOTH , FORCST , NLS , PARAM

References

Box, G.P., and G.M. Jenkins (1976), *Time Series Analysis: Forecasting and Control*, Holden-Day, New York.

Ljung, G.M., and G.E.P. Box (1978), "On a Measure of Lack of Fit in Time Series Models", *Biometrika*, Vol. 66, pp. 297-303.

Purpose

Creates a vector of log likelihoods or fitted values for an autoregressive integrated moving average process.

Format

z = ARIMA (*y, d, phi, theta*);
OPLIST = *progopts ;*

Input

y	literal, *N*x1 vector of time series.
d	scalar, degree of differencing.
phi	literal, *P*x1 AR coefficient vector, or scalar zero.
theta	literal, *Q*x1 MA coefficient vector, or scalar zero.
progopts	literal, optional program options

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The ARIMA (p, d, q) process is given by:

$$\phi(L)(1 - L)^d y_t = \theta(L)\epsilon_t$$

where:

$$\begin{aligned} \phi(L) &= 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p \\ \theta(L) &= 1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q \end{aligned}$$

and where L is the backward shift operator, and d is a non-negative integer.

The coefficients of the ARIMA process are estimated using either ML or NLS. d is an integer constant. When $d = 0$, this becomes the ARMA model. y should be detrended, and have zero mean.

The program control options are specified in *oplist*. The options available are:

CONSTANT/[NOCONST] Specifies whether a constant is to be included. CONSTANT should normally be specified for non-differenced series with non-zero mean, unless the constant is explicitly specified as a parameter.

Both stationary and invertibility conditions need to be satisfied. GAUSSX provides a routine called MROOT, which returns the value of the largest root which must have a modulus less than unity. Consequently, constrained optimization is usually required.

See ARFIMA for details on forecasting, and the "General Notes for Non-Linear Models" under NLS. An example of ARIMA is given in test43.prg.

Example

```
FRML eq1 llf = arima(y1, d, phi1, theta1);
FRML ec1 mroot(phi1) <= .9999;
FRML ec2 mroot(theta1) <= .9999;
PARAM phi1 theta1;
   VALUE = .5 .5;
CONST d;
   VALUE = 1;
ML (p,d,i) eq1;
   EQCON = ec1 ec2;
   OPLIST = constant;
```

In this example, an ARIMA $(p = 1, d = 1, q = 1)$ model is estimated using constrained ML, where the constraints are specified in ec1 and ec2, and where MROOT is a GAUSSX routine for returning the value of the largest root.

Source ARFIMAX.SRC

See Also ARFIMA , ARIMA , MROOT , NLS , VARMA

References

Box, G.E.P., Jenkins, G.M., and Reinsel, G. C. (1994). *Time Series Analysis, Forecasting and Control*, San Francisco: Holden-Day.

Purpose

Creates a vector of log likelihoods or fitted values for an autoregressive moving average process.

Format

z = ARMA (*y, phi, theta*);
OPLIST = *progopts ;*

Input

y	literal, *N*x1 vector of time series.
phi	literal, *P*x1 AR coefficient vector, or scalar zero.
theta	literal, *Q*x1 MA coefficient vector, or scalar zero.
progopts	literal, optional program options

Output

z	*N*x1 vector of log likelihoods (ML).
z	*N*x1 vector of fitted values (NLS).

Remarks

The ARMA (p, q) process is given by:

$$\phi(L)y_t = \theta(L)\epsilon_t$$

where:

$$\begin{aligned} \phi(L) &= 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p \\ \theta(L) &= 1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q \end{aligned}$$

and where L is the backward shift operator.

The coefficients of the ARMA process are estimated using either ML or NLS. When there is no MA component, this becomes the AR model. *y* should be detrended, and have zero mean.

The program control options are specified in *oplist*. The options available are:

CONSTANT/[NOCONST] Specifies whether a constant is to be included. CONSTANT should normally be specified for non-differenced series with non-zero mean, unless the constant is explicitly specified as a parameter.

Both stationary and invertibility conditions need to be satisfied. GAUSSX provides a routine called MROOT, which returns the value of the largest root, which must have a modulus less than unity. Consequently, constrained optimization is usually required.

See ARFIMA for details on forecasting, and the "General Notes for Non-Linear Models" under NLS. An example of ARMA is given in test43.prg.

Example

```
FRML eq1  y1=  arma(y1, phi1|phi2, theta1);
FRML ec1 mroot(phi1|phi2) <= .9999;
FRML ec2 mroot(theta1) <= .9999;
NLS (p,d,i) eq1;
   EQCON = ec1 ec2;
   OPLIST = constant;
```

In this example, eq1 returns the vector of fitted values based on an AR coefficient matrix phi1|phi2 and an MA coefficient theta1. These are estimated using constrained NLS, where the constraints are specified in ec1 and ec2, and where MROOT is a GAUSSX routine for returning the value of the largest root.

Source ARFIMAX.SRC

See Also ARFIMA , ARIMA , MROOT , NLS , VARMA

References Hamilton, J.D. (1994), *Time Series Analysis*, Ch. 11.

Purpose

Creates a vector of log likelihoods for a beta process.

Format

z = BETA_D (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, index of the first shape parameter.
pvec	literal, second shape parameter.

Output

z	Vector of log likelihoods.

Remarks

The beta model is used to estimate duration data; however, for a beta process, $0 \leq y \leq 1$.

The expected value of $shape1_i$ is parameterized as:

$$E(shape1_i) = \exp(indx_i).$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β and *pvec*, are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and the second column taking a value of unity if censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM shape2;
FRML eq0 shape1 = b0 +b1*arrtemp + b2*plant;

1  FRML ex1 llfn =  beta_d(fail, shape1, shape2);
   ML (p,i) eq0 ex1;

2  FRML ex2 llfn =  beta_d(fail~censor, shape1, shape2);
   ML (p,i) eq0 ex2;
```

In example 1, a linear exponential beta model is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a similar estimation when some of the data is censored. In both examples, `fail` takes values in the range {0:1}.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose — Creates vectors using bitwise arithmetic.

Format

```
z = IAND ( x, y );
z = IEQV ( x, y );
z = IOR ( x, y );
z = INOT ( x );
z = ISHFT ( x, s );
z = IXOR ( x, y );
q = RADIX ( x, b );
z = RADIXI ( q, b );
```

Input

x	Nx1 vector.
y	Nx1 vector.
s	scalar, shift parameter.
b	scalar, base.
q	Nx_bitmax matrix.
_bitmax	global scalar, wordlength (default = 16)

Output

z	Nx1 result vector.
q	Nx_bitmax matrix.

Remarks — The bitwise routines allow for logical operations at the bit level.

AND	Bitwise AND of *x* and *y*.
IEQV	Bitwise EQV of *x* and *y*.
IOR	Bitwise OR of *x* and *y*.
INOT	Bitwise complement of *x*.
ISHFT	Bitwise shift of *x* by *s* places. Positive values of *s* shift to the right, negative to the left.
IXOR	Bitwise XOR of *x* and *y*.
RADIX	Convert decimal to base *b*. Converts the Nx1 decimal vector *x* to the *n*x_bitmax matrix of the radix of *x* to the base *b*.
RADIXI	Convert base *b* to decimal. Converts the Nx_bitmax matrix *q* to the base *b* to an Nx1 decimal vector.

The bitwise routines are pure GAUSS code, and are used independently of GAUSSX.

Example

```
library gaussx ;
x = 7;
y = 11;
z = iand(x,y);

z = 3
```

This example evaluates the bitwise AND of 7 and 11.

Source BITWISE.SRC

Purpose

Sets or displays a user specified description associated with a GAUSSX variable.

Format

CATALOG *vname, descript ;*
CATALOG (*options*) *vlist ;*
TITLE = *title ;*

Input

options	optional, print options.
vname	literal, required, variable name.
descript	string, required, description.
vlist	literal, optional, variable list.
title	string, optional, title.

Remarks

The first form of the CATALOG statement associates the description specified in the string *descript* with the GAUSSX variable specified in *vname*. The second form displays the descriptions associated with the variables specified in *vlist*. If *vlist* is not specified, all the vectors currently defined in the current GAUSSX workfile will be displayed

The variables specified in *vname* or *vlist* must exist in the current GAUSSX workfile, otherwise an error will be returned. When a SAVE command is executed, the catalog file is saved with an .fst extension. An OPEN command will read in a catalog file if it exists.

Print options include p – pause after each screen display.

CATALOG requires GAUSS version 3.2 or later. See test02.prg for an example creating a catalog, and test03.prg for displaying a catalog.

Example

```
1.   CATALOG  impt
         Imports from all developing countries.\r
         Source:  World Bank.   ;

2.   CATALOG (p) x1 x2 x3 ;

3.   CATALOG;
        TITLE =  1994 Data Base ;
```

In example 1, a description is specified for the variable `impt`. Note the use of `\r` to create a new line in a string. The second example produces a catalogue of the descriptions for `x1, x2` and `x3`, and pauses (p) after each display. The third example displays the descriptions for the entire current GAUSSX workfile.

See Also

COVA , TITLE

Purpose

Computes the cumulative density function for the specified distribution.

Format

y = CDF (*pdfname, xh, p1, p2, p3*);

Input

pdfname	string, the name of the probability distribution.
xh	*N*x*K* matrix, the upper limits for the specified distribution.
p1	*N*x*K* matrix or scalar, first parameter for the specified distribution.
p2	*N*x*K* matrix or scalar, second parameter for the specified distribution.
p3	*N*x*K* matrix or scalar, third parameter for the specified distribution.

Output

y	*N*x*K* matrix of cumulative probabilities.

Remarks

This procedure returns a matrix of cumulative probabilities for the specified distribution.

See the "General Notes for Probability Density Functions" under PDF.

CDF is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
a = 2; b = 4;
let xh = 1 3 10 20;
y = cdf( f,xh,a,b,0);

  y' =  .5555  .8400  .9722  .9917
```

This example evaluates the cdf for an F distribution with 2 and 4 degrees of freedom, at the values shown in xh.

Source

PDFX.SRC

See Also

CDFI, CDFMVN, PDF, RND, STATLIB

Purpose

Computes the inverse cumulative density function for the specified distribution.

y = CDFI (*pdfname, prob, p1, p2, p3*);

Input

pdfname	string, the name of the probability distribution.
prob	*N*x*K* matrix of probabilities.
p1	*N*x*K* matrix or scalar, first parameter for the specified distribution.
p2	*N*x*K* matrix or scalar, second parameter for the specified distribution.
p3	*N*x*K* matrix or scalar, third parameter for the specified distribution.

Output

y	*N*x*K* matrix of inverse cumulative probabilities.

Remarks

This procedure returns a matrix of inverse cumulative probabilities for the specified distribution. *prob* must lie in the [0 1] interval.

See the "General Notes for Probability Density Functions" under PDF.

CDFI is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
mu = 1; s2 = 1;
let prob = .025 .975;
y = cdfi( normal,prob,mu,s2,0);

  y' =  -.9600   2.9600
```

This example evaluates the inverse cdf for a normal distribution with unit mean and variance, for the values shown in prob.

Source

PDFX.SRC

See Also

CDF, PDF, RND, STATLIB

Purpose

Computes the cumulative density function of the multivariate normal density function (lower tail), using recursive integration.

Format

y = CDFMVN (*xh*, *omega*);

Input

Input

	xh	*K*x1 or *K*x*N* matrix, the upper limits of the *K*-variate normal density function.
	omega	*K*x*K* symmetric, positive definite covariance matrix of the *K*-variate normal density function.
	_cdfmin	global scalar, lower bound (-10).
	_cdftol	global scalar, tolerance to skip a branch (1e-10).
	_cdfpnt	global scalar, print progress. (0)
	_cdford	global scalar, the order of the integration 2, 3, 4, 6, 8, 12, 16, 20, 24, 32, 40. (24)
Output	*y*	*N*x1 vector of probabilities.

Remarks

This procedure returns the orthant probability of an *K*-dimensional multivariate normal density function, evaluated using recursive integration.

The GAUSS function of the same name is used – note however that the second argument is the covariance matrix, as opposed to the correlation matrix.

CDFMVN is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
let xh = 1 2 1;
let omega[3,3] = 1 .8 .6 .8 1 .2 .6 .2 1;
p = cdfmvn(xh,omega);
```

This computes the trivariate normal distribution function over the specified range.

Source CDFMVN.SRC

See Also QDFN, CDFBVN, CDFTVN

Purpose

Computes a hierarchical cluster tree and dendrogram.

Format

CLUSTER (*options*) *varlist ;*
CATNAME = *atomname ;*
METHOD = *method ;*
MODE = *metric ;*
OPLIST = *progopts ;*
ORDER = *weights ;*
TITLE = *title ;*
VALUE = *value ;*

Input

options	optional, print options.
varlist	literal, required, variable list.
atomname	literal, optional, a list of element names.
method	literal, optional, linkage distance method. (SINGLE)
metric	literal, optional, metric mode list (EUCLID).
progopts	literal, program control options.
weights	literal, metric weights.
title	string, optional, title.
value	literal, optional, cutoff values.

Remarks

This procedure creates an hierarchical cluster tree, and optionally graphs the tree - a dendrogram. It can be used in the standard manner of deriving categories across spatial dimensions, or can be used to group observations that are similar based on economic or other characteristics.

CLUSTER produces a table showing the members of each cluster conditional on the cutoff distance specified in *value*. If *value* is small, the "gluing" distance is small, resulting in many clusters with few elements per cluster, while if *value* is large, one is further down the tree, with few clusters, each with many elements. If *value* is a vector, a separate table is produced for each element.

The hierarchical cluster tree algorithm depends crucially on the distance metric used, as well as the method used to define a cluster. The distance metric is defined in *metric*. Define d_{rs} as the distance between vectors x_r and x_s. The available metric modes are:

EUCLID Euclidian distance. $d_{rs}^2 = (x_r - x_s)(x_r - x_s)'$. This is best suited where the data has the same units of measurement over all dimensions (default).

STD Standardized Euclidian distance. $d_{rs}^2 = (x_r - x_s)D^{-1}(x_r - x_s)'$ where D is the diagonal matrix of the variance of the data matrix X.

MAHAL Mahalanobis distance. $d_{rs}^2 = (x_r - x_s)V^{-1}(x_r - x_s)'$ where V is the covariance matrix of the data matrix X. Recommended for when different measurement units are used for different characteristics.

CITY City Block (or Manhattan) metric. $d_{rs} = \sum_{i=1}^{n} \mid (x_{ri} - x_{si}) \mid$.

CHEB Chebyshev metric. $d_{rs} = \max_{i=1,\ldots,n} \mid (x_{ri} - x_{si}) \mid$.

MINK Minkowski metric. $d_{rs} = \left(\sum_{i=1}^{n} \mid x_{ri} - x_{si} \mid^{\rho}\right)^{1/\rho}$. The value of ρ is specified in *weights*.

The cluster definition is defined in *method*; the available linkages are:

SINGLE Single linkage, or nearest neighbour, This uses the shortest distance between objects in two clusters (default).

COMPLETE Complete linkage, or furthest neighbour, This uses the largest distance between objects in two clusters.

AVERAGE Average linkage. This uses the average distance between all pairs of objects in two clusters.

CENTROID Centroid linkage. This uses the distance between the centroids of two clusters.

Each element of a cluster can be given an ID in the dendrogram. This is defined in *atomname*. If a single literal is used, then this becomes the root; the default is Obs. Alternatively, a complete list of element names can be specified – there must be as many names as the current sample.

The program control options are specified in *progopts*. The options available are:

PLOT/[NOPLOT] Specifies whether a dendrogram is produced.

FORECAST = vector of cluster ID for each observation, based on the last cluster table produced. This vector is available in the GAUSSX workspace.

Print options include d —descriptive statistics, p —pause after each screen display, and q —no screen output.

An example of CLUSTER is given in test31.prg.

Example

```
SMPL 1 100;
CLUSTER (p,d,s) age salary sex educ;
   CATNAME =  # ;
   MODE = mahal;
   TITLE =  Socio Economic Cluster ;
   VALUE = .5 1 4;
   OPLIST = plot forecast=clusterid;
PRINT (p) clusterid;
```

This example generates an hierarchical cluster tree for 100 elements, based on four characteristics age, salary, sex and educ. MODE is set to the Mahalanobis metric to account for the differing units of measurement between the four characteristics. A dendrogram (cluster tree) is created by specifying the PLOT option in OPLIST. Three cut-off distances are specified in VALUE, resulting in three tables showing the number of clusters and composition of each at each of the three cut-off distances. A vector called clusterid is created in the GAUSSX workspace that contains the cluster number for each observation, based on a cut-off distance of 4.

See Also TABULATE

Purpose To provide a comment on a GAUSSX command file.

Format *? statement ;*

Input *statement* any statement - text, formulae or comments.

Remarks A ? at any position in a command results in the rest of that line being treated as a comment. These comments do not appear in the output file. For other types of comments, see the examples below.

Example

```
1.   ? OLS y c x1 x2;

2.   OLS z c x1 x2;      ? This is a comment

3.   OLS z c x1 x3;      // This is a GAUSS syntax comment

4.   /*
     This type of comment blocks off a block of text
     COVA x1 x2;
     x3 = rndu(20,10);
     */

5.   @ This is a command file listing comment @ ;

6.   @@  This is an execution time comment ;
```

In example 1, the OLS is not carried out since the ? appears to the left of the command. This comment does not appear in the output file. In example 2, the OLS is carried out, and the comment is ignored. Example 3 shows how a comment is ignored using GAUSS syntax. Example 4 shows how a block of code can be commented out, again using GAUSS syntax.

Example 5 shows the syntax for a comment that is displayed as part of the command file listing, again using standard GAUSS syntax. Example 7 shows how a comment can be generated at execution time - such as when one wants titles on the output listing.

See Also GAUSS

Purpose

Defines constants specified in non-linear formulae used by GMM, FIML, ML and NLS.

Format

CONST *plist ;*
ORDER = *order ;*
SYMBOL = *rootname ;*
VALUE = *values ;*

Input

plist	literal, required, name list.
order	numeric, optional, matrix order.
rootname	literal, optional, element name.
values	optional, starting values.

Remarks

The CONST statement adds the variables in *plist* to the list of GAUSSX parameters, updates the value of the parameters if *values* is specified, and creates global symbols for each parameter in *plist*, initialized at the current value.

Constants must be initialized before estimating an equation in which such constants appear. If VALUE is not specified, each constant in *plist* is given a default value of zero. Unlike coefficients specified in a PARAM statement, constants remain fixed during a non-linear estimation. If *values* is specified, the number of elements must match the number of constants given in *plist*. *values* can also be the name of a vector. Thus following a linear estimation, the coefficient values are stored in a vector called COEFF. These values can be used to set the values for a set of constants by setting VALUE = COEFF. Note however that the number of elements in COEFF must be the same as the number of terms in *plist*.

Example

```
1.   CONST a0 ;

2.   CONST b0 b1 b2;
        VALUE = .3 0 -.2;

3.   OLS y c x1 x2 x3;
     CONST a0 a1 a2 a3;
        VALUE = COEFF;
```

```
4.   aval = rndu(4,3);
     CONST amat;
        SYMBOL = a;
        VALUE = aval;
```

In example 1, a single constant is specified. If a0 had previously been defined as a parameter, it maintains its previous value; if not, its value is set to zero. In the second example, starting values are specified by use of the VALUE option. In example 3, the coefficients from the previous regression are stored as a vector (COEFF); in this case a0 will be given the value of the intercept, a1 the coefficient on x1, etc. Example 4 shows how a matrix of random constants, named a11 to a43, can be created.

See Also ANALYZ, FRML, PARAM

Purpose Computes a copula.

Format *s* = COPULA (*n, cx, rtype*);

Input

n	scalar, number of observations.
cx	*K*x*K* correlation matrix, or scalar correlation coefficient.
rtype	scalar or character, correlation method.

Output

s	*N*x*K* matrix of correlated uniform variates.

Remarks Copulas are functions that describe dependencies among variables, and provide a way to create distributions to model correlated multivariate data. Using a copula, a data analyst can construct a multivariate distribution by specifying marginal univariate distributions, and choosing a particular copula to provide a correlation structure between variables. There are a number of different families of copulas; in this context we use a Gaussian copula.

if *cx* is either a Kendall's or Spearman rank correlation matrix, then the inverse CDF of *s* will have the same correlation structure, irrespective of the distribution chosen. This is known as the Inverse method.

Three correlation methods are available; the method is selected by specifying *rtype*:

[0 or 'p']	Pearson.
[1 or 'k']	Kendall Tau b.
[2 or 's']	Spearman Rank

COPULA is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
let rmat[3,3] = 1 .5 .2 .5 1 .6 .2 .6 1;
q=copula(1000,rmat,1);
v1=normal_cdfi(q[.,1],0,1);
v2=expon_cdfi(q[.,2],2);
v3=gamma_cdfi(q[.,3],1.5,2.5);
```

```
rmat;

     1.0000000       0.50000000       0.20000000
    0.50000000        1.0000000       0.60000000
    0.20000000       0.60000000        1.0000000

corr(v1~v2~v3,1);

     1.0000000       0.49734134       0.19561562
    0.49734134        1.0000000       0.60317918
    0.19561562       0.60317918        1.0000000
```

q is a 1000x3 copula matrix with a Kendall Tau correlation structure given by rmat. This copula is then used to create three correlated random deviates drawn from the normal, exponential and gamma distributions.

Source COPULA.SRC

See Also CORR, MVRND

Purpose

Computes the correlation dimension for a time series or set of vectors.

Format

```
CORDIM (options) vlist ;
    MAXSTEP = step ;
    ORDER  = order ;
    PERIODS = periods ;
    RANGE = range ;
    WEIGHT = wtname ;
```

Input

options	optional, print options.
vlist	literal, required, variable list.
maxstep	literal, optional, number of steps (20)
order	literal, required, embedding dimension
periods	literal, optional, lags. (1 1)
range	literal, optional, range for *r*.

Output

_CORDIM	Correlation dimension.
_CORINT	Vector of correlation integrals.

Remarks

The correlation dimension is the fractal dimension of the phase space of a time series. It is estimated by calculating the separation between every pair of N data points and sorting them into bins of width dr proportional to r, and then estimating the slope of a regression between $\ln(C_m)$ and $\ln(r)$, where C_m, the correlation integral, is the size of each bin.

If vlist contains more than one variable name, it is assumed that each variable represents one dimension of the phase space, and thus the embedded dimension is the number of variables in *vlist*. Otherwise, the minimum number of dynamical variables needed to model the dynamics of the system - the embedded dimension - must be specified in *order*. Similarly, the lag used to reconstruct a phase space from a time series is specified as the first element of *periods*, and the Theiler window for discarding autocorrelated data is specified as the second element of *periods*.

The correlation dimension is given by

$$\lim_{r \to 0} d \ln(C_m) / d \ln(r)$$

Ideally, this should be linear in ln(r), with r as small as possible. A plot of ln(C_m) vs ln(r) is displayed if the g print option is specified. The range for ln(r) is selected automatically, but can be user specified in *range*, and the number of steps can be specified in *maxstep*.

Print options include g —display graphic, p —pause after each screen display, and q —quiet - no screen or printed output. Additional information is available through the on-line help (Alt-H). An example is given in test49.prg.

Example

```
1.  CORDIM (p) x y;

2.  CORDIM (p,g)  z;
       ORDER = 2;
       PERIODS = 1 4;
       MAXSTEP = 40;
       RANGE = -5 -1;
```

In example 1, the correlation dimension is derived for the 2 dimensional state phase given in `x` and `y`. In example 2, the correlation dimension of a time series is investigated, using an embedded dimension of 2, a default time lag of 1, and a Theiler correction of 4 periods. Instead of using the default range of r, a range of ln(r) of -5 to -1 is specified, with 40 steps. A graph of ln(C_m) vs ln(r) is displayed under the g option.

See Also LYAPUNOV

References

Grassberger, P. and I. Procaccia. (1983), “Characterization of Strange Attractors”, *Physical Review Letters* Vol. 50, pp 346-369.

Purpose
Computes a correlation matrix for different correlation types

Format
cx = CORR (*x*, *rtype*);

Input

x	*N*x*K* matrix of data.
rtype	scalar or character, correlation method.

Output

cx	*K*x*K* correlation matrix.

Remarks
This procedure returns the specified correlation matrix.

Three correlation methods are available; the method is selected by specifying *rtype*:

[0 or 'p']	Pearson.
[1 or 'k']	Kendall Tau b.
[2 or 's']	Spearman Rank

CORR is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
cx = corr(x,2);
```

This computes the Spearman rank correlation matrix for the data, x.

Source
COPULA.SRC

See Also
COVA

Purpose

Computes descriptive statistics, covariance and correlation matrix, and for a single variable the autocorrelogram.

Format

COVA (*options*) *vlist ;*
DISPLAY = *screen ;*
FMTLIST = *fmtopts ;*
GROUP = *grouplist ;*
PERIODS = *periods ;*
TITLE = *title ;*
VLIST = *elist ;*
WEIGHT = *wtname ;*

Input

options	optional, print options.
vlist	literal, required, variable list.
screen	literal, optional, screen mode (GRAPH).
fmtopts	literal, optional, format options.
grouplist	literal, optional, group variable list.
periods	numeric, optional, number of lags.
title	string, optional, title.
elist	literal, optional, variable list.
wtname	literal, optional, weighting variable.

Output

MEANS	Vector of means.
STDS	Vector of standard deviations.
MINS	Vector of minimums.
MAXS	Vector of maximums.
SUMS	Vector of sums.
VCOV	Parameter covariance matrix.
VCOR	Correlation matrix.

Remarks

The variables specified in “Outputs” are returned as global variables.

The default print options result in no printed output. Print options includes d —descriptive statistics, c —correlation matrix s —singular value decomposition, v —covariance matrix, *p* —pause after each screen display.

If a single vector is given as the argument list, the correlogram and partial autocorrelogram are displayed if the c option is used. The number of lags used

is 1/3 of the sample size, or *periods*. If DISPLAY = GRAPH, the correlograms are displayed using the PQG screen mode. - see 'General Notes for Graphs" in GRAPH.

Lagged variables can be used by specifying the lag in parenthesis.

Weighting is available using the WEIGHT option. Formatting is available using the FMTLIST option. Grouped output is available using the GROUP option.

Singular value decomposition analysis (SVD) is available using the (s) option. The entire matrix of variables specified in *vlist* must be able to fit in core. Each vector is scaled by the program such that its norm is unity. Variables that are specified as logs should first be e-scaled – this is carried out if the vector is included in *elist*.

Example

```
1.   COVA x1 x2 x3;

2.   COVA (d,v,p) x1 x2 x3(-1);

3.   COVA (p,d,c) x1;
        TITLE =  Analysis of x1 ;

4.   GENR lnx1 = ln(x1);
     COVA (p,s) lnx1 x2 x3;
        VLIST = lnx1;
```

In example 1, the covariance and correlation matrices of the vectors x1, x2, and x3 are returned as global variables; no output is produced. The same analysis is carried out in example 2 but with x3 replaced by x3 lagged once; in this case, descriptive statistics (d) and the covariance matrix (v) are displayed, and execution pauses (p) after each screen display. In example 3, where a single vector x1 is given as the argument list, the correlogram and partial autocorrelogram are displayed since the c option is given. A user specified title is used. In example 4, SVD is carried out (s) on the matrix consisting of the vectors lnx1, x2, and x3. lnx1 is first e-scaled since it is a variable that is measured as a log, and not as a level.

See Also FMTLIST, GROUP, PRINT, TABULATE, TITLE, WEIGHT

Purpose

Creates a vector of log likelihoods for a Cox proportional hazards model.

Format

z = COX (*y, indx, pflag*);

Input

	y	literal, dependent variable - duration.
	indx	literal, index.
	pflag	literal, ties indicator.

Output

	z	Vector of log likelihoods.

Remarks

The Cox proportional hazards model is specified as:

$$H(t, x, \beta) = H_0(t) \exp indx$$

where $H(t, x, \beta)$ is the hazard function, and $H_0(t)$ is the baseline hazard function.

indx is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The log partial likelihood function is defined as:

$$L(\beta) = \sum_{i=1}^{n} \delta \left[f(x_i, \beta) - \ln \sum_{j \in R(y_t)} \exp(f(x_i, \beta)) \right]$$

where $R(t) = \{j | y_j \geq t\}$. The coefficients, β, of the index $f(x_i, \beta)$ are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The Cox model conventionally uses a linear index. For reasonable baseline interpretations, the covariates should be centered so as to have zero mean.

pflag specifies how ties are to be treated. The available methods are:

0	None.
1	Breslow-Peto method.
2	Efron method.
3	Exact (exact marginal-likelihood) method.
4	Discrete (exact partial-likelihood method)

The usual methods are `Breslow` or `Efron`. `Exact` takes considerably longer, and `Discrete` takes so long that it is not recommended. Ties are ignored when *pflag* is set to zero, and the data is used as loaded.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and the second column taking a value of unity if censored, else zero.

Cox residuals are estimated based on the Nelson-Aalen methodology for evaluating the baseline cumulative hazard function. Residuals and survival measures are estimated using the DURATION command.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b1 b2;
FRML eq0 indx = b1*arrtemp + b2*plant;


1  FRML ex1 llfn =  cox(fail, indx, 1);
   ML (p,i) eq0 ex1;

2  FRML ex2 llfn =  cox(fail~censor, indx, 1);
   ML (p,i) eq0 ex2;
   hr = exp(coeff);
   "Hazard Ratio " hr; call keyw;
```

```
3  FETCH fail
   llr0 = sumc(cox(fail,0,1))
   "null model (llr0) " sumc(cox(failuret,0,1));
   call waitkey(1);
```

In example 1, a Cox proportional hazard model with a Breslow methodology for ties is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`.

Example 2 shows a similar estimation when some of the data is censored. The hazard ratio is simply the exponent of the coefficients.

Example 3 shows how one would compute the null model - the likelihood for the baseline hazard function when the index is zero. `llr0`, in conjunction with the likelihood for the full model, can then be used to test the model using the likelihood ratio test.

Source DURATION.SRC

See Also DURATION, ML, NLS

References Hosmer D.W, and S. Lemeshow (1999). *Applied Survival Analysis*, Wiley, New York.

Purpose

To define the size of a new workspace.

Format

CREATE (*options*) *begin end ;*
FNAME = *filename ;*
OPLIST = *progopts ;*

Input

options	(a,q,m,u) optional, the frequency of the data.
begin	literal or numeric, the first period.
end	literal or numeric, the last period.
filename	literal, optional, the name of a GAUSSX save file.
prgopts	literal, optional, options for program control.

Remarks

CREATE must be the first GAUSSX statement in a command file.

There are four data types permitted - Annual, Quarterly, Monthly, and Undated. Undated is the default, unless one uses the format shown in example 5 below, where `gfile` is the name of a GAUSSX save file created in a previous session.

The range specified (eg 1971-1981) must be sufficiently large to accommodate all subsequent operations—that is, a SMPL statement should not go outside these bounds.

The program control options are specified in *progopts*. The options available are:

`DISK/[RAM ]` Specifies the data processing mode. `RAM` is the default, and implies that the GAUSSX workspace is stored in core, while `DISK` implies that the Gaussx workspace is stored on the hard drive, using the pathnames specified in the Project Options screen. The former is faster, while the latter is necessary for huge data sets, such as census data.

A CREATE statement can appear within a command file—this will initialize all data sets. If the command file consists only of pure GAUSS code, then you do not need to put a CREATE statement at the beginning. You should use output on; to switch on that part of the code that you want written to the output file, and wait; or waitkey(1); to pause during execution.

Example

```
1.   CREATE (A) 1971 1981;

2.   CREATE (Q) 19711 19814;

3.   CREATE (M) 197101 198112;
        OPLIST = disk;

4.   n1 = 1; n2 = 20;
     CREATE (U) n1 n2;

5.   CREATE (A);
        FNAME = gfile;
```

In each of the first four examples, a GAUSSX workspace is created; the frequency of the data being annual, quarterly, monthly and undated respectively. In example 3, the GAUSSX workspace is stored on the hard drive, rather than RAM. In example 5, `gfile` is the name of a file previously created by GAUSSX using the SAVE command. The current GAUSSX workspace becomes identical to that at the time the file was saved, and the frequency is specified as annual.

See Also

END, OPEN, NFACTOR, SAVE

Purpose

Creates contingency tables under the current sample.

Format

CROSSTAB (*options*) *varlist ;*
CATNAME = *categories ;*
FMTLIST = *fmtopts ;*
GROUP = *grouplist ;*
MODE = *statmode ;*
TITLE = *title ;*
WEIGHT = *wtname ;*

Input

options	optional, print options.
varlist	literal, required, list of two variable names.
categories	literal, optional, a list of category names.
fmtopts	literal, optional, format options.
grouplist	literal, optional, group variable list.
statmode	literal, optional, statistic mode list (NUM).
title	string, optional, title.
wtname	literal, optional, weighting variable.

Output

STATS Tabular output.

Remarks

The CROSSTAB statement creates a 2-way contingency table, based on the two variables defined in *vlist*. The table is based on the current sample. A total is automatically generated. See TABULATE for full details.

Print options include p —pause after each screen display, and s — print contingency table statistics.

Example

```
CROSSTAB (p,s)  Activity Gender;
    MODE    =  num  fit min max row ;
    CATNAME =  Slight Moderate High Female Male ;
```

A cross-tabulation of Activity against Gender is produced under the current sample.

See Also

FREQ, TABULATE

Purpose

Creates a vector of log likelihoods for a double-bounded dichotomous choice model.

Format

z = DBDC (*r, t, pred, sigma*);

Input

r	literal, matrix of responses.
t	literal, matrix of bids.
pred	literal, utility or wtp vector.
sigma	literal, standard deviation of residual.

Output

z	Vector of log likelihoods.

Remarks

The DBDC coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models. Given the unobserved latent willingness to pay (wtp) variable y^*, and the observed categorical response variable r_1 and r_2 to a first and second question, then the DBDC model is given by:

$$\begin{aligned} y^* &= f(x,\beta) + \epsilon \\ r_1 &= 1 \quad \text{if} \quad y^* > t_0 \\ r_2 &= 1 \quad \text{if} \quad r_1 = 1 \text{ and } y^* > t^u \\ &\qquad \text{or} \quad r_1 = 0 \text{ and } y^* > t^l \end{aligned}$$

$t = \{t_0\ t^u\ t^l\}$ is a three element price vector. $r = \{r_1\ r_2\}$ is an Nx2 response matrix. The first vector, r_1 is a response (yes/no) to whether the wtp exceeds t_0, while the second vector, r_2 is a response to whether the wtp exceeds t^u or t_l, the choice depending on whether the wtp was greater or less than t_0. *pred* is the predicted value ($f(x,\widehat{\beta})$ of the latent variable – the structural equation for the latent variable can be linear or non-linear. ϵ is assumed distributed $N(0,\sigma^2)$.

While *t* is usually fixed for all respondents, there are some circumstances where this is not the case. In such a circumstance, *t* can be specified as an *n*x2 matrix, where the first column is the price offered for the first question, and the second column is the price offered for the second question.

See the "General Notes for Non-Linear Models" under NLS, and the example under ML. An example is given in test35.prg.

Example

```
let t = 5 7 3;
FRML eq1 pred = b0 + b1*sex + b2*educ;
FRML eq2 llf = dbdc(r1~r2,t,pred,sig);
PARAM b0 b1 b2 sig;
   VALUE = 4 0 1 0;
ML (p,i) eq1 eq2;
   TITLE =  DBDC estimation ;
```

In this example, the structural form for the latent variable is shown in (eq1). It can be linear or non-linear. The second equation specifies the likelihood function for the DBDC process. The first question poses a bid value of 5, while the second question poses a value of 7 if the first question was a yes, else poses a value of 3. r1 and r2 are the categorical responses to each question.

Source

GXPROCS.SRC

See Also

ML, NLS

References

Hanemann, W. M., J. Loomis, and B.J. Kanninen (1991), "Statistical Efficiency of Double-bounded Dichotomous Choice Contingent Valuation", *American Journal of Agricultural Economics*, Vol 73, pp.1255-63.

Purpose

To remove the noise from a noisy signal or time series using wavelets.

Format

DENOISE (*options*) *dlist ;*
MODE = *mode ;*
ORDER = *order ;*
PERIODS = *periods ;*
TITLE = *title ;*
VALUE = *value ;*
VLIST = *vlist ;*
WAVELET = *filter ;*

Input

options	optional, print options.
dlist	literal, required, denoised series list.
mode	literal, optional, threshold mode (MAD HARD UNIV).
order	numeric, optional, wavelet level.
period	numeric, optional, exclusion level (5).
title	literal, optional, title.
value	numeric, optional, threshold value.
vlist	literal, required, variable list of original series.
wavelet	literal, optional, wavelet filter (DAUB4).

Values in parentheses are the default values.

Remarks

DENOISE filters noise from a noisy signal or time series. This is accomplished by first deriving the wavelet coefficients, and then thresholding those coefficients below a certain value, since low value coefficients will be associated with noise. The cleaned signal is then reconstructed from the adjusted wavelet coefficients.

The names of the noisy series are given in *vlist*; for each element, a denoised series is estimated and stored in the corresponding name in *dlist*. These vectors can then be used as if they had been created with a GENR statement.

Wavelets Wavelets have different smoothness, symmetry and support properties. The filter family and the number of moments is specified in *wavelet* - the default is DAUB4. High moment filters with larger number of coefficients results in better approximation.

[DAUB] Daubechies orthogonal wavelets— DAUB1 to DAUB10. The Haar wavelet is equivalent to DAUB1.

SYM Symlets wavelets— SYM1 to SYM10. A symlet is more symmetric than the Daubechies wavelet.

COIF Coiflets wavelets— COIF1 to COIF5. Coiflets are less asymmetric than the DAUB or SYM families, and thus have larger support.

Level The order of the wavelet decomposition tree that is produced under the multiple level decomposition is specified in *order*. This is commonly called the level. Since each decomposition uses half the sample at that level, the maximum number of levels, *m*, is determined by the sample length that is exactly divisible by 2^m. The default *order* is zero, which results in the maximum level possible.

Threshold Denoising involves zeroing those wavelet coefficients that fall beneath a specified threshold. The details of how the threshold is established and utilized is determined by the MODE statement, which consists of three components:

Estimate The threshold estimate can be specified directly by the user in *value*, or can be derived using one of the following algorithms:

[UNIV] Universal threshold, $\lambda = \sqrt{2 \ln n}\,\sigma$. This is a global threshold that is asymptotically optimal; it is also smoother than MINIMAX.

MINIMAX The Minimax threshold is a global threshold that is optimal in terms of minimax risk.

SURE The SURE threshold minimizes Stein's unbiased estimator of risk. This method should be implemented with soft-thresholding. This is an adaptive procedure, since the thresholds are determined at each level.

Error The size of the noise in the signal is estimated from the standard deviation of the wavelet coefficients at the finest scale, since this should contain mainly noise. Two estimates are available:

[MAD] Mean absolute deviation from the median. This tends to be more robust.

STD Standard deviation of wavelet coefficients.

Rule Two thresholding schemes are implemented:

[HARD] Coefficients set to zero if absolute value is less than threshold.

SOFT Coefficients set to zero if absolute value is less than threshold; remaining coefficients are shrunk towards zero by the amount of the threshold.

The threshold is applied to all the wavelet coefficients, excluding the 5 (default) coarsest coefficients. The exclusion level can be changed by specifying the number of coefficients to exclude in *periods*.

There must be sufficient workspace for the entire series to be stored in core. Missing values are not permitted.

Print options include p —pause after each screen display, and q —quiet - no screen or printed output.

Examples of the use of DENOISE are given in test38.prg.

Example

```
1.   DENOISE signew;
        VLIST = sigold;

2.   DENOISE (p) inventd;
        VLIST = invent;
        WAVELET = SYM8;
        ORDER = 5;
        MODE =  mad sure soft;
```

In the first example, the original series sigold is denoised to form the new series signew. The default settings imply the use of the Daubechies filter with 4 vanishing moments. The threshold is derived using the Universal estimator with hard thresholding and a MAD estimator of the noise standard deviation. The wavelet level used is maximal.

The second example shows how various options are specified - in this case, the level is set to 5, the filer to Symlet 8, and the threshold is derived using

the SURE estimator with soft thresholding and a MAD estimator of the noise standard deviation.

References

Donoho, D. and I. Johnstone (1995), "Adapting to unknown smoothness via Wavelet Shrinkage", *Journal American Statistical Assoc.*, Vol 90, pp 1200-1224.

Vidakovic, B. (1999), *Statistical Modeling by Wavelets*, John Wiley, New York.

Purpose To create a data vector or matrix of a particular type of stochastic process.

Format *y* = DGP *vstruct ;*

Input *vstruct* DGPS structure, required.

Output *y* Data vector or matrix.

Remarks DGP provides a method for creating a data vector or matrix of a particular type of process, either as a GAUSS or GAUSSX command. The only input argument required is a data structure (*vstruct*), which is a structure of type DGPS. For each type of data generating process, only those elements of *vstruct* that are relevant need be specified.

A brief description of each process follows. In each case a structure of the form `struct DGPS vs;` is assumed:

`arch`	vs.index	structural component
	vs.arch	ARCH parameter vector
	vs.process	string: `arch`
`arch_t`	vs.index	structural component
	vs.arch	ARCH parameter vector
	vs.df	degrees of freedom for t distribution
	vs.process	string: `arch_t`
`arfima`	vs.ar	autoregressive parameter vector
	vs.ma	moving average parameter vector
	vs.diff	fractional differencing parameter
	vs.stderr	residual standard error
	vs.constant	process constant
	vs.process	string: `arfima`

arima	vs.ar	autoregressive parameter vector
	vs.ma	moving average parameter vector
	vs.diff	integer differencing parameter
	vs.stderr	residual standard error
	vs.constant	process constant
	vs.process	string: arima
arma	vs.ar	autoregressive parameter vector
	vs.ma	moving average parameter vector
	vs.stderr	residual standard error
	vs.constant	process constant
	vs.process	string: arma
brownian	vs.diff	fractional parameter (Hurst)
	vs.process	string: brownian
garch	vs.index	structural component
	vs.arch	ARCH parameter vector
	vs.garch	GARCH parameter vector
	vs.process	string: garch
garch_t	vs.index	structural component
	vs.arch	ARCH parameter vector
	vs.garch	GARCH parameter vector
	vs.df	degrees of freedom for t distribution
	vs.process	string: garch_t
gaussian	vs.variance	variance
	vs.process	string: gaussian
linear	vs.index	structural component variable list
	vs.variance	residual variance
	vs.vlist	GAUSSX variable list
	vs.process	string: linear

linear_t	vs.index	structural component variable list
	vs.variance	residual variance
	vs.df	degrees of freedom for t distribution
	vs.vlist	GAUSSX variable list
	vs.process	string: linear_t
logit	vs.index	structural utility variable list
	vs.prob	% data in alternative #1 (binomial only)
	vs.variance	disturbance variance vector or matrix
	vs.process	string: logit
poisson	vs.index	structural component ($\ln \lambda$)
	vs.process	string: poisson
probit	vs.index	structural utility variable list
	vs.prob	% data in alternative #1 (binomial only)
	vs.variance	disturbance variance matrix
	vs.process	string: probit
tobit	vs.index	structural component
	vs.stderr	residual standard error
	vs.process	string: tobit
wiener	vs.process	string: wiener

General Notes

SYNTAX The structural component will normally consist of a variable name (or names) - for example `let vs.index = xb`. However, if DGP is being called as part of a GAUSS command, then the index can be set as to a global variable - for example, `vs.index = 10+20*x1`.

BROWNIAN If `diff`, the Hurst parameter, is not specified, a standard Brownian process is generated. Otherwise, a fractional Brownian process is generated for $0 < diff < 1$.

GARCH Arch and Garch processes automatically generate the conditional variance as a GAUSS global stored under the name `_ht`.

LINEAR For the linear model, a single equation is implied if a single index variable is specified. For the normally distributed error, either vs.stderr or vs.variance must be specified, while for the t_distribution, vs.df is required.
A multivariate normal or t distribution is implied if there is more than a single index variable - each variable represents the structural form for that equation. Both distributions require a residual variance specified in vs.variance, while for the t distribution, vs.df is required. This DGP returns an endogenous variable for each equation, and so the GENR statement cannot be used in this particular case. The results are returned in the variable list specified in vs.vlist.

QR For the logit and probit processes, binomial logit and probit is implied if a single index variable is specified. vs.prob is the mean probability of alternative 1. The logit and probit models assume the disturbances are distributed with a Weibull or normal distribution respectively, which requires scaling by specifying either vs.scale (logit), vs.stderr or vs.variance. This DGP returns a categorical vector with elements of 0 and 1.
A multinomial logit or probit distribution is implied if there is more than a single index variable - each variable represents the structural utility associated with that alternative. For MNL, only the diagonal elements of vs.variance are used. This DGP returns a categorical vector with elements {1..*k*}, where *k* is the number of alternatives.

A number of examples are given in test07.prg, test43.prg and test44.prg.

Example

```
1.   struct DGPS gs;
     gs.arch = .4 |.15 ;
     gs.garch = .3;
     let gs.index = xb;
     gs.process =  garch ;
     GENR y = dgp(gs);

2.   struct DGPS qrs;
```

```
     GENR xb = 4 +5*x1-3*x2;
     let qrs.index = xb;
     qrs.prob = .4;
     qrs.stderr = 1;
     qrs.process =  probit ;
     GENR y = dgp(qrs);

3.   struct DGPS qrls;
     GENR x0 = 0*c;
     GENR x1 = 2-4*z1;
     GENR x2 = 3+5*ln(z3);
     let qrls.index = x0 x1 x2;
     qrls.scale = .25;
     qrls.process =  logit ;
     GENR ycat = dgp(qrls);

4.   let vmat[2,2] = .5 .2 .2 .8;
     struct DGPS ls;
     let ls.index = xb1 xb2;
     ls.variance = vmat;
     let ls.vlist = y1 y2;
     ls.process =  linear ;
     call dgp(ls);
     PRINT (p) y1 y2;
```

The first example demonstrates the creation of a vector y consisting of a structural component $(10 + 2 * x)$ and a residual with a garch distribution, with the parameters for the arch process specified in `gs.arch` (the first element is the constant), and the parameters for the garch process specified in `gs.garch`.

The second example shows how a binomial process is specified using DGP; 40% of the generated data will fall in category 1.

The third example demonstrates a multinomial logit DGP, with 3 alternatives.

Example 4 shows how a linear system is generated with correlated error structure. The structural components (the RHS) are in `xb1` and `xb2`, and the

endogenous variables - y1 and y2 - are created in the GAUSSX workspace, and subsequently printed.

Source DGPX.SRC

See Also GENR, RND

Purpose

To compute an aggregate price index from several underlying price series.

Format

DIVISIA *pindx qindx ;*
VLIST = *vlist ;*

Input

pindx	literal, required aggregate price index.
qindx	literal, required aggregate quantity index.
vlist	literal, required pairs of price and quantity vectors.

Remarks

DIVISIA computes a Divisia aggregate price and quantity index from several underlying price and quantity series. These are chain-linked Laspeyres Indices – the current price is used as the base for estimating the rate of growth to the next period.

The index is derived by calculating the weighted sum of the rates of change of component prices. The weights are calculated using the geometric average of the expenditure shares. The index is normalized to unity at the beginning of the sample. Once the price index *pind* has been determined, the quantity index *qind* is derived by dividing total expenditure by the price index.

The elements in *vlist* consist of the price and quantity of each of the component vectors; the price vector is given first, then the quantity vector. There must be sufficient workspace for the entire matrix to be stored in core. Missing values are not permitted.

Example

```
DIVISIA pindx qindx;
   VLIST = pa qa pb qb pc qc;
```

A Divisia price index `pindx` and quantity index `qindx` is created from three underlying goods `a`, `b`, and `c`; note that the price and quantity are expressed in pairs, with the price first.

See Also

PRIN, SAMA

References

Jorgenson, D., and Z. Griliches (1971), "Divisia Index Numbers and Productivity Measurement", *Review of Income and Wealth*, Vol. 17(2), pp. 227-229.

Purpose To remove the specified variables from the current GAUSSX workspace.

Format DROP *vlist ;*

Input *vlist* literal, required, variable list.

Remarks The DROP statement deletes the specified variables from the GAUSSX workspace. The current SMPL remains in effect.

Example

```
DROP x1 x2 x3;
```

The variables `x1, x2` and `x3` are deleted from the GAUSSX workspace.

See Also KEEP, RENAME, STORE

Purpose

Creates seasonal dummy variables, or a set of dummy variables for a categorical variable.

Format

DUMMY *dname ;*
VLIST = *vname ;*

Input

dname literal, required, root name of dummies.
vname literal, optional, categorical variable.

Remarks

The DUMMY command takes the categorical variable *vname*, and evaluates the total range of dummies required, which are then created using *dname* as a base. The dummies are numbered sequentially, starting at 1.

Seasonal dummy variables are automatically created when the VLIST option is not specified; the number of dummies is determined by the type of workspace defined in the CREATE statement.

If more than 12 dummies are created from a categorical variable, and if these dummies are subsequently used in a GENR or FRML statement, then these variables must be initialized using the GAUSS clear statement.

Example

```
1.   CREATE (q) 19741 19804;
     DUMMY sd;

2.   DUMMY ed;
        VLIST = educ;
```

In the first example, 4 seasonal dummies (sd1, sd2...) are created. In the second example, assume that educ is a categorical variable taking values 1,2,3 or 5, depending on the level of education, then this command will form five dummy variables - ed1, ed2, ed3, ed4 and ed5. ed4 will be a vector of zeros.

Purpose

Computes residuals, survival and hazard rates, based on the last duration model estimation.

Format

DURATION *varlist ;*
BOUND = *level ;*
MODE = *oplist ;*
VALUE = *value ;*
VLIST = *vist ;*

Input

varlist	literal, required, variable list.
level	numeric, optional, percentage confidence level. (.95)
mode	optional, duration measure.
value	literal or numeric, probability value.
vlist	literal, parameter list.

Remarks

The DURATION command computes residuals, survival or hazard rates, where the standard error and confidence bands are based on the last survival model estimation. Duration models typically model the duration of an event, or the time to failure. The following survival models are supported:

BETA_D	— Beta distribution
COX	— Cox model
EXPON	— Exponential distribution
GAMMA_D	— Gamma (distribution) process
GOMPERTZ	— Gompertz process
GUMBEL	— Gumbel (largest extreme value) process
INVGAUSS	— Inverse Gaussian process
LOGISTIC	— Logistic process
LOGLOG	— Loglogistic process
LOGNORM	— Lognormal process
NORMAL	— Normal process
PARETO	— Pareto process
PEARSON	— Pearson process
SEV	— Smallest extreme value process
WEIBULL	— Weibull process

varlist consists of up to four elements - the statistic, the standard error, and the lower and upper confidence bands. The survival measure bands are lower truncated at zero. If *varlist* consists of less than four elements, then only these elements will be evaluated.

The duration measure is set in *mode*. Let f and F be the duration model pdf and cdf respectively, and y the duration. The available survival measures are:

CUMFAIL	The cumulative failure rate. $CHF = F(y)$.
CUMHAZARD	The cumulative hazard rate. $CHZ = -\ln(1 - F(y))$.
HAZARD	The hazard rate. $HZ = f(y)/(1 - F(y))$.
INVSURV	The inverse survival rate. $ISV = F^{-1}(1 - p)$.
SURVIVAL	The survival rate. $SV = 1 - F(y)$. (Default).
BASECUMHZD	The baseline cumulative survival rate (COX only).
BASEHZD	The baseline hazard rate (COX only).
BASESURV	The baseline survival rate (COX only).

The available residual measures are:

COXSNELL	The Cox-Snell residual. $-\ln(1 - F(y))$.
DEVIANCE	The deviance residual.
MGALE	The martingale residual. $1 - \ln(1 - F(y))$
RESID	The estimation residual. $y - indx$ or $\ln(y) - indx$.
SCALEDSCH	The scaled Schoenfeld residuals.
SCHOENFELD	The Schoenfeld residuals.
SCORE	The score residuals.
STDRES	The standardized residual. $RESID/\sigma$.

For RESID and STDRES, $\ln(y)$ is used for the exponential, loglog, lognormal and Weibul distributions. To be meaningful, this measure usually requires the index to be a location measure. For INVSURV, p is the probability, and is specified in *value*.

The SCALEDSCH, SCHOENFELD and SCORE residuals require a parameter list corresponding to each covariate; this is specified in *vlist*. As opposed to the other residual measures, these return a residual for each covariate. Thus, if

there are *k* covariates, then *varlist* should have *k* elements. For these residuals, the index must be linear.

No output is returned, but the duration measures are available for use in the same way as a variable created through LOAD or GENR. Note that the PIT (probability integral transformation) test can be used on the survival rate to ascertain whether the hypothesized distribution can be rejected.

Examples are given in test57.prg.

Example

```
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;
FRML eq1 llfn = lognorm(failuret,indx,scale);
FRML ec1 scale >= 0;
PARAM b0 b1 b2;
   VALUE = 15 0 0;
PARAM scale;
   VALUE = 1;

ML (p,i) eq0 eq1;
   EQCON = ec1;

1.  DURATION hz hzerr hzlb hzub;
      MODE = hazard;
    PRINT (p) failuret hz hzerr hzlb hzub;

2.  DURATION isv isverr;
      MODE = invsurv;
      VALUE  = .25;
    PRINT (p) failuret isv isverr ;

3.  DURATION sv;
      MODE = survival;
    TEST (p) sv;
      METHOD = ppcc;
      MODE = uniform;

4.  DURATION rescn rescnerr;
```

```
      MODE = coxsnell;
GENR lfail = ln(failuret);
PRINT (p) lfail rescn rescnerr ;

5.  FRML cq0 indx =  b1*arrtemp + b2*plant;
    FRML cq1 llfn =  cox(failuret,indx,2);
    ML (p,i) cq0 cq1;
    DURATION schv1 shcv2;
       MODE = schoenfeld;
       VLIST = b1 b2;
```

A lognormal duration model is estimated using constrained ML - since the scale parameter must be positive. In the first example, the hazard rate, its standard error and lower and upper bound are derived for each observation, and then printed.

In the second example, the inverse survival function for a probability of 0.25 is derived for each observation, along with the standard error.

The third example shows how the survival rate is evaluated, and then tested for a uniform distribution using the TEST command. A rejection of a uniform distribution implies that the specified distribution (in this case log normal) is rejected.

In the fourth example, the Cox-Snell residual and its standard error are evaluated.

The fifth example shows how the Schoenfeld residuals are estimated following a Cox regression. The parameters association with each of the covariates, *b1* and *b2* are specified in vlist; thus, since there are two covariates, there will be two items in varlist - *schv1* and *schv2*, corresponding to *artemp* and *plant* respectively.

Techical Notes

The formulae do not show the effect of censoring.

See Also

BETA_D, COX, EXPON, GAMMA_D, GUMBEL, INVGAUSS, LOGISTIC, LOGLOG, LOGNORM, NORMAL, PARETO, SEV, SURVIVAL, WEIBULL

Purpose

Creates a vector of log likelihoods for an EGARCH process.

Format

z = EGARCH (*resid*, *bvec*, *gvec*, *pvec*, *mvec*);

Input

resid	literal, vector of residuals.
bvec	literal, $p + 1$ vector of parameters for GARCH process.
gvec	literal, q vector of parameters for lagged error process.
pvec	literal, 3 element vector (θ_0, γ_0, ν)
mvec	literal, 2 element vector of EGARCH–M parameters (m_0, m_1)

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The structural coefficients and the coefficients of Nelson's exponential GARCH or EGARCH process are estimated using maximum likelihood; The generalized form for the EGARCH (p, q) process is:

$$\begin{aligned} y_t &= f(x_t, \beta) + \epsilon_t \\ \epsilon_t &\sim \sqrt{h_t} v_t \\ \ln h_t &= \beta_0 + \sum_{i=1}^{p} \beta_i \ln h_{t-i} + \sum_{j=1}^{q} \gamma_j (\theta_0 v_{t-j} + \gamma_0 [|v_{t-j}| - \mathrm{E}|v_t|]) \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. This structural component should be first estimated using NLS to determine reasonable starting values. The residuals from this process are the first argument (*resid*) to the EGARCH command.

The second equation specifies the distribution of the residuals – v_t is assumed generalized error distributed (GED); this includes the normal as a special case, along with many other distributions with both fatter and thinner tails. When he tail parameter $\nu = 2$, v_t is standard normal; for $\nu < 2$, v_t has thicker tails than the normal, while for $\nu > 2$, v_t has thinner tails than the normal.

The third equation specifies the structural form of the log of the conditional variance $\ln h$. The β are the weights for the lagged $\ln h$ terms; this is the GARCH process in $\ln h$. The second argument to the EGARCH command (*bvec*) contain the *p+1* elements of β, including the constant, β_0.

The γ are the weights for the lagged disturbance v terms; the third argument to the EGARCH command (*gvec*) are the q elements of γ. Three additional parameters are specified in *pvec*; these are θ_0, γ_0, and the tail parameter ν. It is typically the value of θ_0 that permits the asymmetric volatility that is captured in the EGARCH model. Note that for the $q = 1$ case, there is an identification problem - this is solved by specifying γ_1 as a constant equal to unity.

The final argument, *mvec*, relates to the EGARCH–M process. The conditional variance can be introduced into the structural equation by adjusting the residual:

$$\epsilon_t \rightarrow \epsilon_t - m_0 h_t^{m_1}$$

The two parameters, m_0 and m_1, are specified in *mvec*. A scalar zero is acceptable for no conditional variance in the structural equation.

The residuals must be specified in a first FRML, and then the EGARCH process is specified in a second FRML. The EGARCH model is very sensitive to starting values, and can easily blow up. Use OLS and GARCH to get sensible starting values.

The conditional variance, h_t, is available, and stored in each iteration under the global _HT, and can thus be retrieved using the FETCH command.

See the "General Notes for Non-Linear Models" under NLS, and the remarks under ARCH. An example is given in test07.prg.

Example

```
PARAM  c0 c1;
   VALUE = .5 .5;
FRML eq0 y = c0 - c1*x1 ;
NLS eq0;
PARAM  b0  b1 the0 gam0;
   VALUE =    1 .001 .001 .001;
   LOWERB = -10 .001   -2   -2;
   UPPERB =  10 .999    2    2;
CONST g1 nu m0 m1
   VALUE = 1 2 0 1;
FRML eq1 e = y - c0 - c1*x1 ;
FRML eq2 llfn = egarch(e, b0|b1, g1, the0|gam0|nu, m0|m1);
ML (p,d,i) eq1 eq2;
   METHOD = nr bhhh nr;
   TITLE  =  egarch model ;
```

In this example, a linear EGARCH(1,1) model is estimated, with a disturbance assumed distributed normal. The form of the conditional variance for this specification is:

$$\ln h_t = \beta_0 + \beta_i 1 \ln h_{t-1} + \theta_0 \frac{\epsilon_{t-1}}{\sqrt{h_{t-1}}} + \gamma_0 \left[\frac{|\epsilon_{t-1}|}{\sqrt{h_{t-1}}} - \sqrt{2/\pi} \right]$$

Initial structural coefficients are derived using NLS. The residuals are specified in `eq1`, and the log likelihood is returned from `eq2`. Note the parameter restrictions to ensure that the variance remains positive - `b1` in particular should be constrained. (Another way of doing this is to use the `EQCON` statement.) `nu` has been set to 2, implying that the disturbance is normal, and since it is a (1,1) process, `g1` is set to unity.

Source

GARCHX.SRC

See Also

ARCH, GARCH, MGARCH, ML, NLS

References

Engle, R.F., and V.K. Ng (1993), “Measuring and Testing the Impact of News on Volatility”, *Journal of Finance*, Vol. 48(5), pp. 1749-1778.

Nelson, D.B. (1993), “Conditional Heteroskedasticity in Asset Returns: A New Approach”, *Econometrica*, Vol. 59(2), 1993, pp. 347-370.

Purpose To delineate the end of a command file.

Format END *;*

Remarks The END command must be the last statement in the command file. Any commands following the END command are ignored. If an END statement is not specified, the file will be read to the end-of-file.

Example

```
END ;
```

See Also CREATE

Purpose Parameter constraints for non-linear estimation.

Format GAUSSX COMMAND *vlist ;*
EQCON = *cnstrntlist ;*

Input *vlist* literal, required, variable list.
cnstrntlist literal, required, list of constraint equations.

Remarks It is often necessary to estimate the parameters of a non-linear equation system subject to a set of parameter constraints. These constraints can be simple boundary conditions - for example requiring a parameter to be greater than zero, or they can consist of relatively complex nonlinear relationships.

Each constraint is specified as a logical relationship within a type 3 FRML command – see FRML for details. Within the estimating procedure, the constraints are activated by specifying the constraint equations within an EQCON option. EQCON can be used in any non-linear equation system; thus this permits the imposition of non-linear parameter constraints under FIML, GMM, ML and NLS.

Example

```
FRML eq1 y1 = a0 + a1*x1 + b1*x2;
FRML eq2 y2 = b0 + b1*x1 ;
FRML cq1 b1 >= 0;
FRML cq2 b0^2 + 2*b1 <= 2.4;
PARAM a0 a1 b0 b1;
NLS (p,i) eq1 eq2;
   EQCON = cq1 cq2;
```

In this example, a constrained NLS estimation is carried out over the system of equations eq1 and eq2. The constraints are shown in two ways. First, the coefficient of x2 in the first equation is to be the same as the coefficient of x1 in the second. This occurs easily by specifying the same parameter name - b1. Two other constraints are required - first that b1 is non-negative, and second a relationship between b0 and b1. These are shown as logical constraints in cq1 and cq2 respectively. The two constraints are imposed by specifying the equation names in the EQCON option.

See Also FIML, FRML, GMM, ML, NLS

Purpose

Substitutes macro code for non-linear equations.

Format

GAUSSX COMMAND *vlist ;*
EQSUB = *macrolist;*

Input

vlist	literal, required, variable list.
macrolist	literal, required, list of macro equations.

Remarks

Often non-linear equations have common terms, and rather than writing out the full term for each equation, it is more convenient to use a macro to represent the common term. The macro is assigned in a FRML command, and the substitution occurs in the EQSUB option. EQSUB can be used as a macro for creating a matrix in a non-linear process, for imposing parameter restrictions, or as a macro for common terms.

EQSUB can be used in any non-linear equation system, as well as in non-linear FORCST and SOLVE. FORCST assumes that EQSUB is the same as in the preceding estimation methodology, unless EQSUB is explicitly specified. SOLVE requires an EQSUB if macros are used. EQSUB should not be used in dynamic forecasts.

If an EQSUB occurs in an estimation procedure, GAUSSX will generate the macros specified before estimating the residuals for the current iteration. In the FRML definition, note that the macros are assigned (:=) a value; this is necessary for GAUSSX to distinguish macros from ordinary non-linear equations.

Example

```
FRML es1a qq := y-b0*x;
FRML es1b qq := y^2;
FRML es2 gama := sqrt((a1|a2|a3)'(a1|a2|a3));
PARAM a1 a2 a3 b0;
FRML eq1 qq = (a1 + a2*z + a3*z^2)/gama;
NLS (i,d) eq1;
   EQSUB = es1a es2;
NLS (i,d) eq1;
   EQSUB = es1b es2;
```

In this example, two NLS regressions occur. In each, it is required for the norm of the underlying parameters in eq1 be unity. This is specified in equation es2. In the first regression, the macro qq is replaced by the formula given in equation es1a, and in the second by es1b. In each the macro gama is replaced by the formula in equation es2.

See Also FIML, FORCST, GMM, ML, NLS, SOLVE

Purpose Executes a string consisting of a set of GAUSS expressions.

Format EVAL (*str*);

Input *str* string, GAUSS expression.

Remarks The EVAL statement evaluates the string *str* as if the contents of *str* had been typed in at the GAUSS prompt. This facility permits GAUSS to interact with external applications by sending a set of GAUSS commands as a string.

EVAL requires GAUSS 4.0 or higher.

EVAL is pure GAUSS code, and is used independently of GAUSSX.

Example

```
library gaussx ;
str =  x = 14; x^2;  ;
eval(str);
```

This sets **x** as a global variable with a value of 14, and displays 196.

Source GXPROC.SRC

Syntax xmat, namestr = expand(x,hier,std,vlist)

Input: x is an nxk data matrix, no constant

hier is a the hierarcy code: 0 - linear only 1 - linear and quad 2 - linear and cross 3 - linear and cross and quad

std is the std method: 0 - none 1 - std (0 mean, unit variance) 2 - range -1 to +1

vlist is an kx1 strarray of names, or 0

Output: xmat - standardized matrix namestr - column names

Purpose

Expands a matrix in quadrtic and cross terms.

Format

{ *xmat, namestr* } = EXPAND (*x, hier, std, vlist*);

Input

x	*N*x*K* matrix of data.
hier	scalar, hierarchy code.
std	scalar, standardization method.
vlist	*K*x1 string array of names, or zero.

Output

xmat	expanded standardized matrix.
namestr	column names of the expanded matrix.

Remarks

This procedure expands a matrix up to order two.

Four hierarchy methods are available; the method is selected by specifying *hier*:

0	Linear only.
1	Linear and quadratic.
2	Linear and cross.
3	Linear and quadratic and cross.

Three standardization methods are available; the method is selected by specifying *std*:

0	No standardization.
1	Each column standardized to zero mean and unit variance.
2	Each column standardized to a range of -1 to +1.

EXPAND is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
x = rndn(1000,3);
{xmat, namestr}  = EXPAND(x,1,1,0);
```

This computes `xmat` consisting of 3 linear terms and 3 quadratic terms, each column being standardized to zero mean and unit variance. `namestr` will be a string array of length 6, with elements of the form $X1 * X1$.

Source STEPWISE.SRC

See Also XPAND

Purpose

Creates a vector of log likelihoods for an exponential process.

Format

z = EXPON (*y, indx*);

Input

y	literal, dependent variable - duration.
indx	literal, scale index

Output

z	Vector of log likelihoods.

Remarks

The exponential model can be used to estimate duration data. The expected value of $scale_i$ is parameterized as:

$$E(scale_i) = \exp(indx_i).$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the “General Notes for Non-Linear Models” under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn =  expon(fail,indx);
   ML (p,i) eq0 eq1;

2  FRML eq2 llfn =  expon(fail~censor,indx);
   ML (p,i) eq0 eq2;
```

In example 1, a linear Exponential model is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Identify, estimate and forecast the exponential smoothing model.

Format

EXSMOOTH (*options*) *vname ;*
DISPLAY = *screen ;*
MAXIT = *maxit ;*
METHOD = *meth ;*
MODE = *mode ;*
NDIFF = *ndiff ;*
NSDIFF = *nsdiff ;*
OPLIST = *progopts ;*
PERIODS = *periods ;*
RANGE = *range ;*
TITLE = *title ;*
TOL = *tolerance ;*
VLIST = *fcstname ;*

Input

options	optional, print options.
vname	literal, required, variable name.
screen	literal, optional, screen mode (GRAPH).
maxit	numeric, optional, maximum number of iterations (20).
meth	literal, optional, algorithm list (GAUSS GAUSS GAUSS).
mode	literal, optional, smoothing mode (SINGLE).
ndiff	numeric, optional, degree of differencing (0).
nsdiff	numeric, optional, degree of seasonal differencing (0).
progopts	literal, optional, options for program control.
periods	numeric, optional, number of lags for correlogram (15).
range	numeric, optional, pairs of ranges for forecasting.
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
fcstname	literal, optional, forecast variable name.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
LLF	Log likelihood.

VCOV	Parameter covariance matrix.
_PSTAR	Vector of smoothing constants.

Remarks

The EXSMOOTH command undertakes all three parts of the exponential smoothing process - identification, estimation and forecasting. The process is given by:

$$x_{t+1} = f(x_t, \hat{x}_t, \hat{T}_t, \hat{F}_t, \theta)$$

where:

x_t	=	actual value of x at period t.
$\hat{x}_t$	=	forecast value of x for period t.
$\hat{T}_t$	=	forecast value of trend for period t.
$\hat{F}_t$	=	forecast value of seasonality for period t.
θ	=	coefficient vector.

The exponential smoothing algorithm is specified in *mode*. The available algorithms are:

SINGLE	Single exponential smoothing. No trend or seasonality. A single coefficient (α) is estimated. (default).
DOUBLE	Brown's double exponential smoothing; this involves single exponential smoothing carried out twice. No trend or seasonality. A single coefficient (α) is estimated.
HW	Holt-Winters exponential smoothing. This uses trend, but no seasonality, and two coefficients are estimated (α, β).
HWADD	Holt-Winters exponential smoothing with additive seasonal. This uses trend and seasonality, and three coefficients are estimated (α, β, γ).
HWMULT	Holt-Winters exponential smoothing with multiplicative seasonal. This uses trend and seasonality, and three coefficients are estimated (α, β, γ).

The exponential smoothing process requires that *vname* must be in core, and uses the current sample, which must be contiguous.

Print options include *c* —print correlogram of the estimated residuals, *d* —print descriptive statistics, *i* —print parameters at each iteration, *p* —pause after each screen display, and *q* —quiet - no screen or printed output. Additional information is available through the on-line help (Alt-H).

The program control options are specified in *progopts*. The options available are:

IDENTIFY/[NOIDENT] Specifies whether the identification process is to be undertaken.

[ESTIMATE]/NOEST Specifies whether the model is to be estimated.

[FORECAST]/NOFORCST Specifies whether the forecast process is to be undertaken.

[FIT]/RESID Specifies the type of forecast mode.

STATIC/[DYNAMIC] Specifies whether the actual or predicted values of *vname* are used in the forecast process.

PARAM/[NOPARAM] Specifies whether the parameter starting values are to be given in a PARAM or a CONST statement, or if they are to be evaluated by the program.

[PLOT]/NOPLOT Specifies whether the correlogram and partial autocorrelogram are to be plotted.

Identification

The identification process is required to determine the degree of differencing necessary to generate a series that is stationary. This works in exactly the same way as for ARIMA. Normally, differencing is not undertaken in an exponential smoothing context.

Estimation

The estimation process can be used to optimally choose the smoothing constants based on the minimum sum of squared one-step forecast errors. This recursive estimation is handled automatically by GAUSSX . The estimation process requires the user to specify the type of algorithm to be used in *mode*. GAUSSX will automatically estimate starting values of the parameters of the model. These parameters are called ALPHA for the main smoothing parameter, BETA for the trend parameters and GAMMA for the seasonal parameter. If the option OPLIST = PARAM is specified, starting values for the coefficients must be given by the user in a PARAM or a CONST statement. Thus, if some

of the parameters are to be restricted during an EXSMOOTH estimation, they should be specified previously in a CONST statement.

The estimation use the NLS routines, and all the non-linear options are available. Initial conditions follow Chatfield (1978), and residuals are set to zero for these initial observations. GAUSSX uses the current sample, and automatically drops the first $ndiff + freq * nsdiff$ observations if differencing is specified. The frequency used for the seasonal factor is determined by the type of data set specified in the CREATE command. Parameter values at the end of the estimation are stored both under their individual names, as well as in a global vector called COEFF. In addition, the entire vector of smoothing constants is stored in _PSTAR. A correlogram of the residuals is produced if the c option is specified in *options*.

Forecasting

A separate forecast is undertaken for each pair of sample dates specified in *range*, or for the last 15 observations if RANGE is not specified. Under the default (DYNAMIC), the forecasts are based on the actual values of *vname* up to the first element in the pair, and forecast values up to the last element of the pair. Forecasts based on the actual residuals derived during the estimation process can be achieved by using the STATIC option. The vector that is forecast is the fitted value of *vname*, unless OPLIST = RESID is specified, in which case the forecast is the vector of residuals. The forecast for the last pair of sample points specified in *range* is stored as a GAUSSX vector under the name given in *fcstname*.

Forecast values for an EXSMOOTH process can also be obtained using the FORCST command. Both the MODE and the RANGE options must be specified. See the "General Notes for Non-Linear Models" under NLS. Examples of exponential smoothing estimation are given in test16.prg.

Example

```
1.   SMPL 1956 1974;
     EXSMOOTH (p,d) y;
        OPLIST =  noforcst;

2.   SMPL 19681 19854;
     EXSMOOTH (p) q;
        MODE  = hwadd;
```

```
        RANGE = 19841 19874;
        VLIST = qfit;

3.  SMPL 1962 1988;
    PARAM alpha beta
        VALUE = 0.6 0.3;
    CONST beta;
    EXSMOOTH (p) gnp;
        MODE   = hw;
        OPLIST = param identify static resid;
        MAXIT  = 40;
```

Example 1 shows how the smoothing constant for single exponential smoothing is undertaken on the vector y. Summary description of y is undertaken using the (d) option, and the estimation is carried out, but no forecast.

Example 2 shows the more usual case. Here, quarterly data is to be dynamically forecasted starting in 1984.1, based on smoothing constants optimally chosen using the Holt-Winters additive seasonal algorithm. The forecast is stored as the variable qfit, and can be used in subsequent GAUSSX operations.

A restricted model is estimated in Example 3: β is restricted to 0.3 through the previous CONST statement; while α takes a starting values of 0.6. The identification process is undertaken, and after estimation the static residuals for the last 15 observations is displayed, but not saved.

See Also ARIMA, CONST, FORCST, NLS, PARAM, TITLE

References Chatfield, C. (1978), "The Holt-Winters forecasting procedure", *Applied Statistics*, Vol. 27, pp. 264-279.

Newbold, P., and T. Bos (1990), *Introductory Business Forecasting*, South-Western Publishing Co. Cincinnati, Ohio.

Purpose

To fetch the named vectors and store them as global variables.

Format

FETCH *varlist ;*
VLIST = *matname ;*

Input

varlist literal, required, variable list.
matname literal, optional, matrix name.

Remarks

The FETCH command instructs GAUSSX to access the named vectors, and store them as global variables. The current SMPL statement remains in effect. This command allows the user to use the named series in subsequent GAUSS commands. The VLIST option allows the user to place the GAUSSX variables directly into the matrix *matname*. See Appendix C for details on how GAUSSX treats variables in the GAUSSX workspace.

Example

```
1.   library pgraph;
     SMPL 1 20 ;
     FETCH x;
     BAR(0,x);

2.   FETCH  z1 z2 z3;
        VLIST = z;
     FETCH y yhat;
     e = y - yhat;
     ehhe = e'z*inv(z'z)*z'e;
     @@  ehhe   ehhe;
```

The first example shows how other types of graphs, that are available in GAUSS can be used in GAUSSX – in this case a bar graph. The second example shows how one could use the matrix capability of GAUSS to derive estimators which are functions of the data, again from within a GAUSSX command file.

See Also

GAUSS, STORE

Purpose

Evaluates a type II nonlinear FRML statement, and stores the result in the GAUSSX workspace.

Format

FEVAL *varlist ;*
EQNS = *eqnlist ;*

Input

varlist literal, required, variable list.
eqnlist literal, required, equation list.

Remarks

The FEVAL statement evaluates each equation in *eqnlist* as if the statement had been rewritten in a GENR statement, and stores the result in the respective member of *varlist* in the GAUSSX workspace. The elements of *varlist* do not have to be the same as the elements on the LHS of each equation.

Example

```
FRML eq1 y1 = a0 + a1*x1 + a2*ln(x2);
FRML eq2 y2 = b0 + b1*x1 + b2*x2;
FEVAL yh1 yh2;
   EQNS = eq1 eq2;
GENR x1 = x1^2;
FEVAL y1;
   EQNS = eq1;
```

In this example, each equation (eq1 and eq2) is evaluated, and the result stored in yh1 and yh2 respectively. After the GENR statement, eq1 is reevaluated, and the result stored in y1.

See Also

GENR

Purpose

Creates a vector of log likelihoods for a fractionally integrated GARCH process.

Format

z = FIGARCH (*resid, avec, bvec, gvec*);
z = FIGARCH_T (*resid, avec, bvec, gvec, dvec*);

Format

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for ARCH process.
bvec	literal, vector of parameters for GARCH process.
gvec	literal, dimension parameter.
dvec	literal, distributional parameter (ν).

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The structural coefficients and the coefficients of the FIGARCH process are estimated using maximum likelihood. The FIGARCH model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ h_t &= \alpha_0 + \Theta(L)\epsilon_t^2 + \sum_{j=1} \beta_j h_{t-j} \end{aligned}$$

where:

$$\begin{aligned} A(L) &= \alpha_1 L + \alpha_2 L^2 + ... + \alpha_q L^q \\ B(L) &= \beta_1 L + \beta_2 L^2 + ... + \beta_p L^p \\ \Phi(L) &= [1 - A(L) - B(L)](1 - L)^{-1} \\ \Theta(L) &= 1 - B(L) - \Phi(L)(1 - L)^d \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies

the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The β are the weights for the lagged h terms; this is the GARCH process. Both the α and the β terms enter as weights for the lagged squared residual.

The first element of *avec*, which is required, gives the constant. *gvec* is the dimension parameter (d) for the FIGARCH process; normally this parameter should lie between zero and unity. A value close to zero implies a long memory process, while a value close to unity implies a very short memory.

Note the stationarity conditions described under GARCH. In addition, the elements of Θ should be positive to ensure non-negative conditional variance. This can usually be ensured by requiring that $d + \alpha_1 > 1$

See the "General Notes for GARCH" under GARCH, and the "General Notes for Non-Linear Models" under NLS.

Example

```
OLS y c x1 x2;
sigsq = serˆ2;
PARAM c0 c1 c2;
   VALUE = coeff;
PARAM a0 a1 a2 b1 d1;
   VALUE =  sigsq .1 .1 0 .95;
FRML cs1 a0 >= .000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 b1 >= 0;
FRML cs5 a1 + d >= 1;
FRML cs6 d >= 0;
FRML cs7 d <= 1;
FRML eq1 resid = y - (c0 + c1*x1 + c2*x2);
FRML eq2 lf = figarch(resid,a0|a1|a2,b1,d1);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4 cs5 cs6 cs7;
```

In this example, a linear FIGARCH model is estimated using constrained maximum likelihood, with OLS starting values. The residuals are specified in eq1,

and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive.

Source GARCHX.SRC

See Also GARCH, EQCON, FRML, ML, NLS

References Baillie, R.T, T. Bollerslev, and H.O. Mikkelsen. (1996), "Fractionally integrated generalized autoregressive conditional heteroskedasticity", *Journal of Econometrics*, Vol 74, pp 3-30.

Purpose Filters data using a variety of filters.

Format *y* = FILTER (*ftype, x, p1, p2, y0*);

Input

ftype	string, the name of the filter.
x	*N*x*K* matrix, data to be filtered.
p1	*P*x*K* matrix or scalar, first parameter for the specified filter.
p2	*Q*x*K* matrix or scalar, second parameter for the specified filter.
y0	*P*x*K* matrix or scalar, initial values.

Output *y* *N*x*K* matrix of filtered data.

Remarks The available filters are:

`ARMA` *y* = filter("*arma*", *x, phi, theta, y0*);

$$(1 - \phi(L))y = (1 + \theta(L))x$$

This is an autoregressive moving average filter. `phi` is the AR component, and `theta` is the MA component. The first elements of `y` are pre-specified with `y0`, which should have the same order as `phi`.

`DEDIFF` *y* = filter("*dediff*", *x* ,*d, 0, 0*);

This is an inverse difference filter, where `d` is the integer degree of differencing.

`DETREND` *y* = filter("*detrend*", *x, 0, 0, 0*);

This is a detrending filter - `y` has zero mean, and is detrended.

`DIFF` *y* = filter("*diff*", *x, d, 0, y0*);

$$y = (1 - L)^d x$$

This is a difference filter, where `d` is the degree of differencing - both integer and fractional differencing are supported. The first

elements of y are pre-specified with y0, which should be of order ceil(d).

hp *y* = filter(*"detrend", x, w, 0, 0*);

This is the Hodrick-Prescott filter, where w is a parameter that controls the trade off between fit and smoothness. w = 0 implies that y has the same trend component as the original series. Suggested values for w are 100 for annual data, 1600 for quarterly, and 14400 for monthly data.

LINEAR *y* = filter(*"linear", x, a, b, y0*);

$$y_n = b_1 * x_n + b_2 * x_{n-1} + \cdots + b_{nb+1} * x_{n-nb} - a_1 * y_{n-1} - \cdots - a_{na} * y_{n-na}$$

This is a one dimensional recursive digital filter. The data in vector x is filtered by vectors a and b to create y. The linear filter is an IIR (infinite impulse response) or recursive filter. Initial conditions are specified in y0.

STANDARD *y* = filter(*"standard", x, 0, 0, 0*);

This filter creates the standardized series y with zero mean and unit variance.

FILTER is pure GAUSS code, and is used independently of GAUSSX.

Example

```
library gaussx;

1.   let a = .5 .3 .1;
     let b = .2 ;
     let y0 = 0 0 0;
     y = FILTER( linear,  x, a, b, y0);

2.   d = 1;
     xd = FILTER( diff', x, d, 0, x[1]);
     x2 = FILTER( dediff,  xd, d, 0, 0);
```

Example 1 shows a linear filter - in effect an AR(3) process, but with the current and one period lag value of **x** being part of the process. Example 2 shows a first order differencing, followed by its inverse.

Source FILTERX.SRC

Purpose

Estimates the coefficients of a non-linear equation or system of equations using full information maximum likelihood.

Format

FIML (*options*) *elist ;*
BOUND = *level ;*
EQCON = *cnstrnt ;*
ENDOG = *endlist ;*
EQSUB = *macrolist ;*
IDENT = *ilist ;*
JACOB = *Jacobian ;*
MAXIT = *maxit ;*
MAXSQZ = *maxsqz ;*
METHOD = *meth ;*
POSDEF = *pdname ;*
SIMANN = *simann ;*
STEP = *step ;*
TITLE = *title ;*
TOL = *tolerance ;*
TRUST = *trust ;*
WEIGHT = *wtname ;*

Input

options	optional, print options.
elist	literal, required, equation list.
level	numeric, optional, percentage confidence level.
cnstrnt	literal, optional, list of constraint equations.
endlist	literal, required, endogenous variable.
macrolist	literal, optional, macro equation list.
ilist	literal, optional, identity list.
Jacobian	literal, optional, Jacobian.
maxit	numeric, optional, maximum number of iterations (20).
maxsqz	numeric, optional, maximum number of squeezes (10).
meth	literal, optional, algorithm list (GAUSS GAUSS BHHH).
pdname	literal, optional, positive definite algorithm (NG).
simann	numeric, optional, SA options (5 .85 100 20).
step	literal, optional, step type (LS).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).

	trust	numeric, optional, TR options (.1 1 .001 3).
	wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
LGCOEFF	Vector of Lagrangian coefficients.
GRADVEC	Gradient vector.
LLF	Log likelihood.
VCOV	Parameter covariance matrix.
COVU	Residual Covariance matrix.

Remarks

The FIML command estimates the coefficients of a single equation or a system of equations by iteratively choosing those parameters which maximize the likelihood function. FIML differs from NLS only in that it takes into account the Jacobian, and thus allows the estimation of equations in which true simultaneity exists. Although GAUSSX will evaluate the Jacobian if it is not specified, given the endogenous variables, this increases the computation time considerably.

See the "General Notes for Non-Linear Models" under NLS, and the examples given in test02.prg and test03.prg.

Example

```
PARAM a0 a1 a2 b0 b1 b2 b3;
FRML eq1 y1 = a0 + a1*x1 + a2*y2;
FRML eq2 y2 = b0 + b1*x3 + b2*x4 + b3*y1;
FRML eq3 y2 = a0 + b1*x3 + b2*x4 + b3*y1^2;

1.   FIML eq1 eq2;
        ENDOG = y1 y2;
        MAXIT = 10;
        JACOB = 1  -a2
                -b3  1;

2.   FIML (p,i) eq1 eq3;
```

```
ENDOG  = y1 y2;
TOL    = .0001;
METHOD = nr gauss bhhh;
JACOB  =   1     -a2
         -2*b3*y1  1;
```

In the first example, a linear system of equations is estimated by FIML; note how the Jacobian is written—GAUSSX takes care of evaluating this, and notes that it does not change across cases. In the second example, the Jacobian is a function of the variables, and has to be evaluated for each observation. The initial method is changed from GAUSS (the default) to NR, and parameter values are printed at each iteration (i), with a pause (p) after each screen display.

See Also FRML, GMM, ML, NLS, TITLE, WEIGHT

References Amemiya, T. (1985), *Advanced Econometrics* Harvard University Press, Cambridge.

Purpose

Creates a vector of log likelihoods for a multinomial probit process, without parameterization of the covariance matrix.

Format

z = FMNP (*ycat, vmat*);

Input

ycat literal, Nx1 vector of alternatives chosen (not ranked), or *N*x*K* matrix or rankings of alternatives chosen. (ranked)

vmat literal, matrix of utility values for each alternative.

_cmnpup global scalar, update _vcmat every _cmnpup iterations (Default = 2),

_cmnpck global scalar, carry out check for improving llf after vcmak: 0 - false, 1 - true. (Default = 1).

FMNP uses MNP to evaluate the multivariate normal integral, and thus the following globals are used - see MNP for documentation.

_mnpsc global scalar, scaling option.

_mnpint global scalar, integration algorithm.

Output

z Vector of log likelihoods.

Remarks

The structural coefficients and the coefficients of the FMNP process are estimated using maximum likelihood.

ycat is an *N*x1 vector in which is specified the alternative chosen for each observation. If ranked data is available, *ycat* is an *N*x*K* vector in which is specified the ranking for each alternative for each observation. Each utility is specified in a FRML, and since utility differences are evaluated, the first utility is set to zero as a reference; *vmat* is the matrix formed by the concatenation of these utilities. The utilities can be functions of individual characteristics, (multinomial probit), choice characteristics (conditional probit), or a combination, and can be linear or non-linear.

The standard MNP procedure evaluates the probability of selecting the alternative specified in *ycat*. For each observation, the mean value (utility) associated with each alternative is stored in *vmat*. The Random Utility Model

assumes that the utilities are distributed with the specified mean, and an additive disturbance that is correlated across alternatives. In the MNP formulation, the distribution of these errors is multivariately normal, with a covariance matrix Σ. For a K alternative model, there are $K^* = .5K(K-1)$ possible two choice combinations. Under FMNP, this covariance matrix is simulated based on the current structural parameter values and knowledge of the alternative actually chosen, or the ranking of alternatives in the RMNP case. Integration of the multivariate density function is undertaken by QDFN. Exact estimation is the default, and is acceptably rapid for low K, or for the factor analytic case. For large K, simulation methods using the GHK algorithm are utilized. The QDFN globals must be set before a MNP estimation.

Since the utilities are estimated as differences, a reference is needed; usually this is achieved by setting the first utility equal to zero. Similarly, the scaling of the parameters is determined by the covariance matrix – in the FMNP estimation, the norm of the K^* covariance elements is set to unity.

See the "General Notes for Non-Linear Models" under NLS, and the discussion of multinomial probit MNP. An example is given in test33.prg.

Example

```
FRML zp1 v1 = 0;
FRML zp2 v2 = g0 + g1*x1 + g2*x2;
FRML zp3 v3 = h0 + h1*x1 + g2*x3;
FRML zpmnp lllf = fmnp(ycat,v1~v2~v3);
PARAM g0 h0 g1 g2 h1;
ML (p,i) zp1 zp2 zp3 zpmnp;
   METHOD = nr nr nr;
TITLE =  Non-linear FMNP ;
```

In this example, a linear mixed FMNP model is estimated. `x1` is a individual characteristic, while `x2` and `x3` are choice based characteristics. `ycat` should take values of 1, 2, or 3, depending on which alternative was selected for each observation.

Source FMNP.SRC

See Also ML, MNP, NLS

References Breslaw, J.A. (2002). "Multinomial Probit Estimation without Nuisance Parameters", *Econometrics Journal*, Vol. 5(2), pp. 417-434.

Purpose

User control over formatted input for packed ASCII OPEN and formatted output for COVA, PRINT and ASCII SAVE.

Format

GAUSSX COMMAND *vlist ;*
FMTLIST = *fmtopts ;*

Input

vlist literal, required, variable list.
fmtopts literal, required, options for format control.

Remarks

The format control options are specified in *fmtopts*. The options available are:

FORMAT = *fmtstring* where *fmtstring* is either a single string as used in the GAUSS format command, and applies to all the variables being printed, or is the name of a *k*x1 vector containing the desired format string for each of the *k* variables in *vlist*. Acceptable strings are RD, RE, RO, RZ, LD, LE, LO, LZ.

RECORD = *recordwidth* where *recordwidth* is a scalar giving the length of the packed ASCII record, excluding the final carriage return and line feed.

POSITION = *position* where *position* is either a scalar giving the position for the first variable, or is the name of a *k*x1 vector containing the position for each of the *k* variables in *vlist*.

WIDTH = *fieldwidth* where *fieldwidth* is either a scalar giving the width for each variable, or is the name of a *k*x1 vector containing the desired widths for each of the *k* variables in *vlist*.

PRCN = *precision* where *precision* is either a scalar giving the precision for each variable, or is the name of a *k*x1 vector containing the desired precision for each of the *k* variables in *vlist*.

An option that is not specified is set to the default value. If the width is not sufficiently large for the number to be displayed, it will be ignored for that observation. Note also that FMTLIST is a subcommand, and applies only to the preceding main command.

Example

```
1.   PRINT x1 x2 x3;
        FMTLIST =  WIDTH=10 PRCN=4;

2.   let fvec = RD RD LE LO;
     let wvec = 10 10 12 8;
     COVA (p,d) x1 x2 x3 x4;
        FMTLIST =  FORMAT=fvec  WIDTH=wvec PRCN=0;

3.   let pvec = 4 12 23;
     let wvec = 2  8  3;
     let dvec = 0  2  0;
     OPEN x1 x2 x3;
        FNAME = data.asc;
        FMTLIST =  RECORD=36 POSITION=pvec WIDTH=wvec
                   PRCN=dvec;
```

In the first example, a width of 10 characters is reserved for each variable, with 4 digits displayed after the decimal point.

In the second example, each of the variables is displayed with formats specified in fvec, field widths specified in wvec, and with the decimal point suppressed.

In the third example, three fields are read from a packed ASCII file with record length of 36 (excluding final carriage return and line feed). x1 occurs in columns 4-5, x2 in 12-19 and x3 in 23-25. No adjustment for decimal points is made for x1 or x3. For x2, a decimal point is inserted two places in from the right edge of the field. x1 and x3 may have decimal points in the data, but x2 must not have any, nor may any element of x2 be missing.

See Also

COVA, OPEN, PRINT, SAVE

Purpose

Forecasts or compute variables based on the last estimation, or on a user specified procedure.

Format

FORCST *varlist ;*
BOUND = *level ;*
ENDOG = *endoglist ;*
EQNS = *eqnlist ;*
EQSUB = *macrolist ;*
METHOD = *meth ;*
MODE = *mode ;*
RANGE = *rangelist ;*
USERPROC = *&userprc ;*
VALUE = *values ;*
VLIST = *vlist ;*

Input

varlist	literal, required, variable list.
level	numeric, optional, percentage confidence level. (.95)
endoglist	literal, optional, endogenous variable list.
eqnlist	literal, optional, equation list.
macrolist	literal, optional, macro equation list.
meth	literal, optional, estimation method.
mode	literal, optional, type of forecast (FIT STATIC).
rangelist	numeric, optional, pairs of ranges for forecasting.
&userprc	literal, optional, pointer to user procedure.
values	literal or numeric, optional, coefficient values.
vlist	literal, optional, input variable list.

Remarks

The FORCST computes a forecast for the dependent variable(s) associated with the most current estimation. This can be for a single equation, or for a system of equations. In the default, the forecast is static, but if there are lagged dependent variables, a dynamic forecast can be specified for most estimation methods. FORCST is not designed to estimate forecast values when any of the RHS variables is a current endogenous variable; for this case, use the SOLVE command. Forecasts can be based on the historical value of the RHS, or on future expected values of the RHS variables. The FORCST command can also be used to create variables that are created in a

user specified procedure, as well as predicted value and standard errors for variables that are non-linear functions of estimated parameters

The type of forecast can be set by the MODE options. The available estimation modes are:

STATIC	Lagged dependent variables take their historical values. This is the default except for ARIMA and EXSMOOTH where the both *mode* and *rangelist* must be specified.
DYNAMIC	Lagged dependent variables take their simulated values.
NAIVE	Naive step ahead forecast (ARFIMA, ARIMA, ARMA only).
BLP	Best linear predictor step ahead forecast (ARFIMA, ARIMA, ARMA only).

The type of forecast output is also determined by the MODE option. The available output modes are:

[FIT]	The fitted value of the dependent variable(s).
RESID	The residuals of the estimation.
RESIDSQ	The square of the residuals of the estimation.
STDERR	The standard error of the forecast(OLS, ARFIMA, ARIMA, ARMA only).
LLF	The log likelihood (ML only).
CONDVAR	The conditional variance (GARCH models only)
BOUNDS	Prediction limits (OLS only).
COOK	Cook's D (OLS only).
DFFITS	Scaled difference in fitted value (OLS only).
DFBETAS	Scaled difference in coefficients (OLS only).
HAT	The Hat vector ($x_i(X'X)^{-1}x_i'$) (OLS only).
STDRES	Standardized residuals (OLS only).
STUDENT	Studentized residuals (OLS only).
MILLS	Mill's ratio (QR only).
PROB	Probability forecast for each alternative (MNL, MNP, PROBIT and LOGIT only).
CAT	Category forecast (MNL, MNP, PROBIT and LOGIT only).

No output is returned, but the forecast variables are available for use in the same way as a variable created through LOAD or GENR. In most cases, none of the suboptions need to be specified; the estimation methodology and the list of equations are taken from the previous estimation. Note that the number of elements in *varlist* must equal the number of equations in the last estimation. For MODE = DFBETAS the number of elements in *varlist* must equal the number of coefficients in the last estimation.

The various options provides a degree of flexibility in using FORCST. These are required if a previous estimation has not been carried out, or if one wishes to evaluate a forecast in a different context than the previous estimation. Examples are given in test02.prg, test07.prg, test16.prg, and test19.prg.

Example

```
1.   FORCST yfit;

2.   FORCST y1fit y2fit;

3.   OLS y c x1 x2 y(-1);
     FORCST  residft;
        MODE = resid dynamic;

4.   OLS y c x1 x2;
     FORCST lb ub;
        MODE = bounds;
        BOUND = .9;

5.   QR eq1;
     FORCST yfit;
        METHOD = OLS;
        EQNS   = eq2;

6.   FORCST xfit;
        METHOD = EXSMOOTH HW;
        MODE   = DYNAMIC;
        RANGE  = 1980 1992;
        VALUE  = .5 .3;
```

```
7.   aval = rndu(3,5)-0.5;        ? random coefficients
     proc sigmoid(x);
        retp(1./(1+exp(-x*aval)));
     endp;
     LIST avlist av1 av2 av3 av4 av5;
     FORCST avlist;
        VLIST = c x1 x2;
        USERPROC = &sigmoid;


8.   PARAM b0 b1 b2 sigma;
        VALUE = 1 1 1 .5;
     FRML eq1 qhat  =  b0*(K^b1).*(L^b2);
     NLS (p) eq1;
     FRML eq2 q2 = exp(ln(b0) + (b1+b2)*ln(Z));
     FORCST q2;
       EQNS = eq2;
     FORCST q2se;
       EQNS = eq2;
       MODE = stderr;
```

In the first example, a single equation has been estimated previously. `yfit` is the forecast value of the endogenous variable for the current sample; if the sample is the same as existed during the regression, then `yfit` shows the tracking of y. Note that the forecast is estimated on the structural form of the equation.

In the second example, a two equation system was last estimated; `y1fit` and `y2fit` are the predicted values for the respective LHS variables.

In example 3, the dynamic residuals from the last regression are stored in `residft`.

Example 4 shows how a 90% confidence band is derived for the fitted values - the prediction limit.

In example 5, a the coefficients from a previous QR estimation are used to

predict eq2, using an OLS methodology.

Example 6 shows an exponential smoothing forecast using the Holt-Winters algorithm with α = 0.5 and β = 0.3; xfit will contain the historical values of x until 1980, and a dynamic forecast from 1981 - 1992.

Example 7 shows a user specified forecast; *userproc* takes the matrix of the vectors specified in VLIST as its argument, and returns the matrix specified in avlist. Hence x is the nx3 matrix of c, x1, and x2, and sigmoid(x) returns an nx5 matrix, whose columns are stored under the names av1, av2..av5.

In example 8, a Cobb-Douglas production function is estimated using NLS. Predicted values are derived in q2 using the first forecast, and the standard error in q2se for each observation in the second forecast.

See Also ARIMA, EXSMOOTH, QR, SOLVE

References Belsley, D., E. Kuh, and R. Welsch (1980), *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*, John Wiley and Sons, New York.

Purpose — Creates a vector of log likelihoods for a frontier production function model.

Format — *z* = FPF (*resid, s, lam*);

Input

resid	literal, vector of residuals.
s	literal, standard error of residuals.
lam	literal, ratio of the standard errors.

Output

z	Vector of log likelihoods.

Remarks — The frontier production function coefficients are estimated using maximum likelihood. Given a production function, $f(x,\beta)$, the model is given by:

$$y = f(x,\beta) + \epsilon$$
$$\epsilon = \nu - \mu$$

Thus the residuals from the production function, ϵ, consist of two components, ν and μ, where ν is $N(0,\sigma_\nu^2)$, and $\mu \geq 0$. The model can be estimated by determining two parameters, *s*, the standard error of ϵ, and *lam*, the ratio of σ_μ to σ_ν.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test41.prg.

Example

```
OLS y c x;
PARAM s; value = ser;
PARAM b0, b1; value = coeff;
PARAM lam; value = .1;
FRML eq1 resid = y - b0 - b1*x;
FRML eq2 lf = fpf(resid,s,lam);
ML eq1 eq2;
```

In this example, OLS is used to get starting values of the structural coefficients `b0, b1` and the standard error of the residuals `s`.

Source GXPROCS.SRC

See Also ML, NLS

References Madalla, G.S (1983), *Limited-Dependent and Qualitative Variables in Econometrics*, Cambridge University Press, pp. 194-196.

Purpose

Computes frequency distributions under the current sample.

Format

FREQ (*options*) *vname ;*

Input

options optional, print options.
vname literal, required, variable name.

Remarks

The FREQ statement estimates the frequency distribution for the variable defined in *vname*. The counts are based on the current sample.

Print options are p —pause after each screen display.

Example

```
1.   FREQ (p) x1 ;

2.   TABULATE (p) c;
        VLIST = x1;
```

A frequency distribution of x1 is produced under the current sample. The descriptive statistics are not printed. An equivalent procedure using the TABULATE command is shown in the second example.

See Also

CROSSTAB, TABULATE

Purpose

Defines a formula which can then be used in subsequent estimations.

Format

(1) FRML *fname vlist ;*
(2) FRML *fname vname = formula ;*
(3) FRML *fname formula reop value ;*
(4) FRML *fname vname := formula ;*

Input

fname	literal, required, equation name.
formula	literal, required, formula.
value	numeric, required, RHS value.
reop	literal, required, relational operator.
vlist	literal, required, vector list.
vname	literal, required, LHS variable name.

Remarks

Each of the estimation procedures used by GAUSSX requires a knowledge of the structural form to be estimated; this is defined in one, or a number of FRML commands. There are four types of formulae; type I—linear, type II—nonlinear; type 3 — nonlinear relational; and type 4 —non-linear macro definition. Linear formulae are used by AR, ARCH, OLS, POISSON, QR, SURE, VAR, 2SLS, and 3SLS, while non-linear formulae are used by FIML, GMM, ML, NLS, and SOLVE.

Type I equations are used in linear estimations; they consist of a unique equation name, followed by a vector list. The first element in the list is the LHS variable, while the remaining elements are the RHS variables (VAR is an exception). Lags are shown as in example 2 below. Care should be taken to make sure that the lagged variables exist in the current sample. Thus, if the sample is 1974 to 1981, and one uses `x(-2)`, then data must exist for `x2` for 1972 and 1973.

Type II equations are the structural equations used in non-linear estimation and SOLVE; they consist of a unique equation name, followed by a non-linear equation written in the form: LHS = formula. Note that for non-linear equations, any parameters or constants must be defined (PARAM, CONST) before the equation is estimated. The syntax of *formula* is standard GAUSS. Thus although `a0*x1` is acceptable when `a0` is a parameter (scalar), the element by element rules must be used for vector operations i.e. `x1.*x2`. See the remarks in GENR.

Type 3 equations consist of non-linear parameter relationships that are used as constraints in non-linear estimation. The syntax consists of the form:

formula reop value

where *reop* is one of the three relational operators, <=, ==, >=. The syntax of *formula* is standard GAUSS , and *value* is a numeric value. Each of these formulae are specified as parameter constraints using the EQCON option during a non-linear estimation.

Type 4 equation are macro definitions. They are written exactly like Type II equations, except that they are written in the form:

mname := *formula*

mname is the name of a macro, and *formula* is the value that will be substituted in place of the macro when an EQSUB option is encountered. See ANN for an example.

Example

```
1.   FRML eq1 y1 c x1 x2;

2.   FRML eq2 y2 c x1 x1(-1) x1(-2) x2;

3.   FRML eq3 y3 = a0 + a1*x1 + a2*x2^a3;

4.   FRML eq4 y4 = a0 + a1*x1 + a2*x2;

5.   FRML cq1 b0 + b1 >= .5;
     FRML cq2 b3 == 0;
     FRML cq3 b0^2 + 2*b1 <= 2.4;

6.   FRML es1 lcost := ln(w*l + v*m + r*k);
     FRML es2 gama := sqrt((a1|a2|a3)'(a1|a2|a3));
```

Type 1 formulae are shown in examples 1 and 2; the LHS variable (y1, y2)

comes first, followed by the list of RHS variables. An intercept is shown by the vector of unity (c). Each equation is identified by a user given name (eq1,eq2). In example 2, lagged values of x1 are specified - the length of the lag being shown in parenthesis.

Type II formulae are shown in examples 3 and 4. Example 3 shows a typical non-linear equation—these are estimated using iterative non-linear procedures which take longer to solve than for the linear case. Nothing stops a non-linear equation actually being linear—as shown in example 4.

Parameter constraints are shown in example 5. The constraints can be linear or nonlinear, and can involve a number of parameters, or simply provide a bound for a parameter.

Macro definitions are shown in example 6. They can simply replace a part of a formula that is used often, or can invoke specific parameter restrictions. The macro substitution occurs at the EQSUB option.

See Also ANALYZ, CONST, EQCON, EQSUB, FEVAL, PARAM

Purpose

Computes the efficient frontier of a portfolio.

Format

FRONTIER *rlist ;*
OPLIST = *progopts ;*
PERIODS = *periods ;*
VLIST = *returns ;*

Input

rlist	literal, required, asset list.
progopts	literal, optional, options for FRONTIER.
periods	numeric, optional, number of points (20).
returns	literal, required, expected return for each asset.

Output

_risk	*m*x1 vector of standard deviations.
_return	*m*x1 vector of rate of return.
_weights	*mxk* weighting matrix.

Remarks

The FRONTIER statement returns uses the Markowitz model to identify the set of efficient portfolios. Efficient portfolios have the lowest aggregate variance for a given yield.

Given the data on the rate of return for each asset over the current sample, the frontier is evaluated for the number of points specified in periods. The future expected return is specified in vlist. If *m* points on the frontier are specified, then FRONTIER returns the *m*x1 vectors of standard deviation (_RISK) and rate of return (_RETURN), as well as the *mxk* weighting matrix (_WEIGHT), where *k* is the number of assets specified.

Print options include d – print descriptive results, and p – pause after each screen display.

The program control options are specified in *progopts*. The options available are:

[PLOT]/NOPLOT	Specifies whether a plot of the frontier is produced.

An example is given in test25.prg.

Example

```
let returns =  6.5 9.3 8.9 8.6;
FRONTIER ibm intel msft hp;
   VLIST = returns;
   PERIODS = 25;
```

This example calculates the efficient frontier at 25 points for the four stocks based on the current sample. A plot of the frontier is also created. The GAUSS variables _RISK, _RETURN and _WEIGHTS are returned, with dimensions 25x1, 25x1, and 25x4 respectively.

Purpose

Calculate the future value of a stream of payments.

Format

y = FV (*pmt, r, n*);

Input

pmt	*n*x1 vector, or scalar, periodic payment.
r	*n*x1 vector, or scalar, interest rate at each period.
n	scalar, number of periods.

Output

y	Scalar, future value of the periodic payments.

Remarks

The FV statement returns the future value of a stream of payments over time. The payment is made at the beginning of each period; thus the first element of *pmt* earns interest in the first period. If *pmt* is a scalar, then the payment stream consists of *pmt* at each period. If *r* is a scalar, then the interest rate is assumed the same over the *n* periods. If *pmt* and/or *r* are vectors, they must have lengths of *n*. Interest rate is per period; thus an annual rate of 9% paid monthly for 20 years would have $r = .09/12 = 0.0075$, and $n = 12 * 20 = 240$.

FV is pure GAUSS code, and is used independently of GAUSSX .

Example

```
library gaussx ;
pmt  = 100;
r    = .1/12;
n    = 120;
fval = fv(pmt,r,n);

fval = 20655.20
```

This calculates the future value of a stream of payments of $100 per month for 10 years, with a discount rate of 10%

Source

FINANCE.SRC

See Also

AMORT, MCALC, PV

Purpose
Creates a vector of log likelihoods for a gamma process.

Format
z = GAMMA_D (*y*, *indx*, *pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, scale index.
pvec	literal, positive shape parameter.

Output

z	Vector of log likelihoods.

Remarks
The gamma model can be used to estimate duration data. The expected value of $scale_i$ is parameterized as:

$$E(scale_i) = \exp(indx_i).$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index and *pvec* are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and the second column taking a value of unity if censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM shape;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML ex1 llfn =  gamma_d(fail, indx, shape);
   ML (p,i) eq0 ex1;

2  FRML ex2 llfn =  gamma_d(fail~censor, indx, shape);
   ML (p,i) eq0 ex2;
```

In example 1, a linear exponential gamma model is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a similar estimation when some of the data is censored.

Source

DURATION.SRC

See Also

DURATION, ML, NLS

References

King, G., J. Alt, N. Burns and M. Laver (1990). "A Unified Model of Cabinet Duration in Parliamentary Democracies," *American Journal of Political Science*, Vol. 34(3) pp. 846-871.

Purpose

Creates a vector of log likelihoods for a GARCH process.

Format

z = GARCH (*resid, avec, bvec*);
z = GARCH_T (*resid, avec, bvec, dvec*);

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for ARCH process.
bvec	literal, vector of parameters for GARCH process.
dvec	literal, distributional parameter (ν).

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The structural coefficients and the coefficients of the GARCH process are estimated using maximum likelihood. The GARCH model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ h_t &= \alpha_0 + \sum_{i=1} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1} \beta_j h_{t-j} \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The α are the vectors of the weights for the lagged ϵ^2 terms; this is the ARCH process. The β are the weights for the lagged h terms; this is the GARCH process. Thus if α is just α_0, and β is zero, we have OLS; if α is a vector, and β is zero, we have standard ARCH; otherwise we have some type of GARCH.

General Notes

Models The following models are supported:

ARCH	Single equation ARCH model.
GARCH	Single equation generalized ARCH model.
AGARCH	Single equation asymmetric GARCH model.

EGARCH	Single equation exponential GARCH model.
FIGARCH	Single equation fractionally integrated GARCH model.
IGARCH	Single equation integrated GARCH model.
MGARCH	Multiple equation multivariate GARCH model.
PGARCH	Single equation power GARCH model.
TGARCH	Single equation truncated GARCH model (GJR).

Formula Structure For all these procedures, the disturbance is typically given in the first (structural) formula, for example:

```
FRML eq1 u = y - s0 - s1*x1 - s2*x2;
```

If a moving average process is required, this can be given in a second formula; thus for an MA1 process: the formula would be:

```
FRML eq2 e = recserar(u,u[1],theta);
```

Finally, the likelihood is given in the third formula

```
FRML eq3 lllf = garch(e, a1|a2, b1);
```

Garch in the Mean Garch in the mean – for example, GARCH-M can be carried out for each of the single equation methods, since the conditional variance, h_t, is available, and stored in each iteration under the global _HT. The structural formula for such a process would be given by:

```
FRML eq1 u = y - s0 - s1*x1 - s2*x2 - thi*sqrt(_ht);
```

Residual distribution For the single equation models, the residuals are assumed distributed normal, with the exception of EGARCH, in which they are assumed to have a generalized error distribution (GED). The Student-t distribution can also be specified by calling GARCH_T, etc. An additional distribution parameter (ν) is required.

Parameter Constraints Garch processes normally require parameter constraints to ensure stationarity and nonnegativity of the conditional variances.

$$\begin{aligned} \alpha_0 &> 0 \\ \alpha_i &\geq 0 \\ \beta_i &\geq 0 \\ \sum_{i=1} \alpha_i + \sum_{j=1} \beta_j &< 1 \end{aligned}$$

These conditions can easily be imposed using the parameter constraint command (EQCON) for non-linear estimation. See type 3 FRML and EQCON for details.

Conditional Variance The conditional variance for all GARCH processes is retrieved using the FORCST command, with MODE = CONDVAR. If no range is specified, the estimated conditional variance based on the actual residuals and estimated parameters is returned. If a range is specified, the estimated conditional variance is returned up to the first date of the range, and the forecast based on the information up to the first date is returned for the period specified.

See the "General Notes for Non-Linear Models" under NLS, and the remarks under ARCH. An example is given in test07.prg.

Example

1.
```
ARCH y c x1 x2;
   ORDER = 1 2;
PARAM g0 g1 g2 a0 a1 a2;
   VALUE = coeff;
PARAM  b1 b2;
   VALUE = .1 .1;
FRML cs1 a0 >= 0.000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 b1 >= 0;
FRML cs5 b2 >= 0;
FRML cs6 a1+a2+b1+b2 <= .999999;
FRML eq1 resid = y - (g0 + g1*x1 + g2*x2);
FRML eq2 lf = garch(resid,a0|a1|a2,b1|b2);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4 cs5 cs6;
```

2.
```
FRML eq1  e = y - g0 - g1*x - thi*sqrt(_ht);
FRML eq2  llfn = garch(e,a0|a1,b1);
PARAM g0 g1 a0 a1 b1 thi;
ML (p,d,i) eq1 eq2;
   METHOD = nr bfgs nr;
   TITLE =  garch-m ;
```

```
     STORE _ht;

3.   FRML eq1  u = y - g0 - g1*x;
     FRML eq2  e = recserar(u,u[1],theta);
     FRML eq3  llfn = garch(e,a0|a1,b1);
     PARAM g0 g1 a0 a1 b1 theta;
     ML (p,d,i) eq1 eq2 eq3;
        METHOD = nr bfgs nr;
        TITLE =  garch (MA1 process) ;
     FORCST hfit;
        MODE = condvar;
        RANGE = 196701 196712;
```

In the first example, a linear GARCH model is estimated, using ARCH starting values. The residuals are specified in eq1, and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive.

In the second example, a GARCH-M process is evaluated. The conditional variance is then stored as a GAUSSX vector.

The third example shows how a GARCH process with an MA1 process for the disturbance is estimated, and how the predicted conditional variance is retrieved.

Source GARCHX.SRC

See Also ARCH, AGARCH, EGARCH, IGARCH, MGARCH, PGARCH, TGARCH, EQCON, FRML, ML, NLS

References Engle, R.F. (1982), “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of the U.K. Inflation”, *Econometrica*, Vol. 50, pp. 987-1007.

Bollerslev, T. (1986), “Generalized Autoregressive Conditional Heteroscedasticity”, *Journal of Econometrics*, Vol. 31, pp. 307-327.

Gouieroux, C. (1997), *ARCH Models and Financial Applications*, Springer-Verlag, New York.

Purpose

Include a GAUSS command within a GAUSSX command file

Format

gausscmnd ;
@@ gausscmnd ;

Input

gausscmnd any legal GAUSS statement.

Remarks

Any GAUSS command can be included in a GAUSSX command file; this is useful, since much of the power of the GAUSS programming language can be directly used by GAUSSX. The examples below show a number of applications; an additional example is shown in FETCH.

The @@ syntax forces output on; and thus should be employed when using the GAUSS Application Modules. Additional information on using the Application Modules under GAUSSX is given in Appendix D.

In some cases it may be necessary to use a GAUSS command which is identical to a GAUSSX command - in this case the @@ syntax identifies this command as a GAUSS command. This syntax will also place a comment in the output file, without the need for an output on; statement.

Example

```
1.   OUTPUT ON;
      This comment will appear in the output file ;
     OUTPUT OFF;

2.   @@  The Variance Covariance matrix is :  VCOV;

3.   goto page1;
       ...
     page1:  ;

4.   i = 1;
       ...
     if i == 1; goto page1; endif;
       ...
     page1: ;
```

```
5.   @@ LOAD x[2,9] = data.asc;

6.   i = 1;
     do until i == 5;
        ...
        i = i + 1;
     endo;
```

Example 1 places a comment on the output file. In example 2, the matrix vcov is printed out with the preceding comment; an output on; statement is not required because of the @@ syntax. An unconditional branch is shown in example 3, and a conditional branch in example 4. In example 5, a matrix is loaded into the variable x; the @@ syntax is needed since LOAD is a GAUSSX command. In equation 6, a do loop is shown; the code within the do loop can be GAUSS or GAUSSX .

See Also COMMENT, FETCH, PAGE, STORE

Purpose Control over the genetic algorithm process.

Format
GAUSSX COMMAND *vlist ;*
METHOD = *method ;*
GENALG = *control ;*

Input
vlist literal, required, variable list.
method literal, required, algorithm list.
control literal, optional, list of control options.

Remarks The genetic algorithm is a search algorithm that attempts to find a global optimum by simulating an evolutionary process. Each member of a population (a chromosome) has a set of genes (parameter values). The members breed, and the genes can mutate. Successful chromosomes - ie those that are most fit, as evaluated by the optimization process - survive, while the rest die.

This algorithm can be very useful for testing if one is at a global optimum, as well as for situations when one gets a "failure to improve likelihood" error message. It is also very robust when parameter upper or lower bounds are encountered.

Genetic algorithm can be implemented as a step method during ML and NLS estimations. Typically, one can use GA for the second element of *method* to find the parameter values, and then use one of the other stepsize algorithms for the final method to evaluate the Hessian. However, it is considerably slower than the other stepsize methods, although the speed can be adjusted by adjusting the control options. GA can be used with constrained optimization - in this case a heuristically determined penalty function is used to constrain the parameters to the feasible region.

Control over the GA options is provided by the GENALG option; this consists of a 4 element vector *control*; these elements are:

1. The population size. Each member (chromosome) of the population has a set of genes which correspond to parameter values. The larger the population, the greater the variability, but the longer the estimation time. Default = 30.

2. Number of matings. While multiple matings result in an increased population, only the original population size is maintained by culling the least fit. Default = 4.
3. Probability of mutation. Genes suffer from mutation, on a chance basis. Default = 0.4.
4. Degree of mutation. Should mutation occur, the degree of mutation is specified with this parameter. Large values make for faster solution, but can result in poorer optimization near the end (but see below). Default = 0.25.

Unlike the other optimization algorithms, GA is not smooth - change occurs sporadically. If there is no change in the best fitness after MAXSQZ iterations, the degree of mutation is reduced by 50%. Convergence is determined by testing when the standard deviation of the fitness is less than the second element of TOL. Since each breeding consists of an iteration, MAXIT should be set fairly high. An example is given in `test21.prg`.

Example

```
NLS(p,i) eq1;
   METHOD = gauss ga nr ;
   GENALG = 25 2 .5 .2;
   MAXIT = 100;
```

This example would undertake non-linear least squares on eq1 using `gauss` as the initial step method, `ga` the remaining steps, except for the final step (where one needs the Hessian) which is estimated using Newton-Raphson (`nr`). The GA process uses a population of 25, with each individual mating twice. Mutation occurs with probability 0.5, and the degree of mutation is 0.2.

See Also ML, NLS

Purpose

To generate a new vector to be created according to a user specified formula, and stored in GAUSSX.

Format

GENR *vname* = *formula ;*

Input

vname literal, required, variable name.
formula literal, required, formula.

Remarks

Data is created only for those cases specified in the current sample. If a vector has previously been defined over a sample space that is longer than the current sample space, then the excluded observations are set to missing value if REPL has been set by the OPTION command – this is the default. The excluded observations remain unaffected if the NOREPL option has been set.

The formulae used can consist of any legal GAUSS expressions. It is thus important that the user understands the GAUSS operations syntax. + and – work as expected; however * and / correspond to matrix multiplication and linear equation solution respectively. Normal element by element multiplication or division uses .* and ./ respectively - see example 2 below, and Chapter 4 of the GAUSS manual.

The user is responsible for making sure that all vectors used on the RHS have previously been defined.

The user is likewise responsible for making sure that the vector operations are conformable, and generate a vector of order $n \times 1$, where n is the current number of observations being read. Normally GAUSS will expand values so that the vectors are conformable – thus the 4 in example 1 is automatically transformed into an $n \times 1$ vector of 4s, to be conformable with x1. Thus note example 4, where pi (π) has to be explicitly multiplied by c to create an $n \times 1$ vector.

C – a vector of unity – can be used in the same way as any other GAUSSX vector. The vector _ID takes all the values implied by the CREATE statement, and can be used as a trend term.

Some GAUSS commands require that the order be specified. Example 6 shows how one deals with this – n is a global variable specifying the current

value for the number of rows.

GAUSSX uses two functions in addition to those specified in the GAUSS manual; these are LAG, which creates a lagged variable, and NMV, (Not Missing Value) which is used in data transformation when missing values exist.

Example

```
1.   GENR y1 = 4 + 2*x1;

2.   GENR y2 = gnp./cpi;

3.   GENR y3 = 5 + 2*lag(y1,1) + 3*lag(y2,2);

4.   GENR y4 = pi*c;

5.   GENR y5 = a0 + a1*x3;

6.   GENR y6 = y5 + rndn(n,1);

7.   GENR y2 = (ln(x1) + abs(x2))^2 ;

8.   GENR smpvec = (y5 .le y6);
```

Example 1 shows a typical GENR operation. Example 2 shows how constant GNP is defined - note the ./ operator. In example 3, the lag operator is shown - `y1` is lagged once, and `y2` is lagged twice. In example 4, note that `pi` is a GAUSS reserved word, and `c` is a GAUSSX reserved word. The series `c` corresponds to a vector of unity – thus `y4` is a vector of π. In example 5, coefficients that have previously been defined in a PARAM or CONST statement are used as part of a formula. Example 6 shows the use of another GAUSSX reserved word – `n`, the number of rows. Example 7 shows the use of GAUSS functions `ln` and `abs`, as well as exponentiation. Example 8 shows how a vector – `smpvec` – can be created which could then be used as a sample selection criteria in a subsequent SMPL statement.

See Also DUMMY, FEVAL, FRML, LAG, LOAD, NMV, NUMDAT

Purpose — Loads a matrix from a binary or ASCII file.

Format — *x* = GETM (*filename*, *k*);

Input

filename	string, name of input file.
k	literal, number of series (cols).

Output

x	*N*x*k* matrix to be read from filename.

Remarks — GETM allows the transfer of matrices from other applications. GETM returns an *N*x*k* matrix *x* from either an ASCII file or a binary data file of type double.

GETM is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
x = getm( c:\temp\mydata.bin, 6);
```

This example reads the matrix x from the binary file c:\temp\mydata.bin.

Source — GXPROCS.SRC

See Also — PUTM

Purpose Calculates Gini coefficients.

Format *g* = GINI (*ymat*);

Input *ymat* *n*x*k* matrix, income data for *k* countries.

Output *g* *k*x1 vector, Gini coefficients.

Remarks The GINI statement returns the Gini coefficient based on an income vector. The vector should consist of incomes for equally distributed income classes. Thus if there are 20 classes, the first would be the mean or median income of the lowest 5%, the second would be for the next 5%, and the 20th for the highest 5%. If *ymat* is a matrix with *k* columns, then each column is taken as a separate country, and *k* coefficients are returned.

GINI is pure GAUSS code, and is used independently of GAUSSX .

Example

```
library gaussx ;
ymat = loadd(income.dat);
g = gini(ymat);
```

Income data is loaded and stored in ymat, The GINI command calculates the Gini coefficient for each column of ymat.

Source GXPROCS.SRC

Purpose Control over the global optimization process.

Format GAUSSX COMMAND *vlist ;*
METHOD = *methlist ;*
GLOBOPT = *ctllist ;*

Input
vlist literal, required, variable list.
methlist literal, required, algoritm list.
ctllist literal, optional, list of control options.

Remarks Global optimization is a search algorithm that attempts to find a global optimum by a direct search over potential optimal hypercubes. It is based on Lipschitzian optimization, but does not require the user to specify the a Lipschitz constant. It is a direct search methodology, that splits the feasible space into ever smaller hypercubes, and continues this process by ascertaining which cubes are potentially optimal. This selection process allows for an efficient evaluation of a high dimensional problem.

The feasible space is determined from the lower and upper bounds that the user specifies for each parameter. The performance of the algorithm is directly related to the size of this space, so the smaller the difference between the lower and upper bounds, the better.

The algorithm is particularly suited to ascertaining a global optimum when a function has many local optima. Traditional hill climbing algorithms will find one of the local optima, and are sensitive to initial starting values. A global search algorithm, by contrast, will search the entire region, and find the global optimum.

Global optimization can be implemented as a step method during ML and NLS estimations. Typically, one can use GO for the second element of *methodlist* to find the parameter values, and then use one of the other stepsize algorithms for the final method to evaluate the Hessian. GO can be used with constrained optimization - in this case a penalty function is used to constrain the parameters to the feasible region.

Control over the GO options is provided by the GLOBOPT option; this consists of a 4 element vector *ctllist*; these elements are:

1. Maximum number of function evaluations. This acts as a basis for initial sizing of the problem, as well as providing a processing limitation. A larger value will allow for more extensive searching, but longer estimation time. Default = 20000.
2. Maximum number of divisions. Each hypercube can be divided and subdivided up to this number of divisions. Default = 100.
3. Jones factor. This is used to determine potentially optimal hypercubes. Experimental values between .01 and 1e-7 seem to work. Default = 0.0001.
4. Stopping criteria. GO is not smooth - changes occur sporadically. To give the algorithm a chance to find better optimum, convergence is not declared until the normal `tol` condition has been satisfied `nstop` times. Default = 4;

Since each hypercube division consists of an iteration, MAXIT should be set fairly high. A number of examples are given in `test51.prg`.

Example

```
CREATE 1 1;
PARAM  x1 x2 ;
  LOWERB = -2 -2;
  UPPERB =  2  2;
FRML eq1
  fcn = (1+(x1+x2+1)^2 .*
        (19-14*x1+3*x1^2-14*x2+6*x1.*x2+3*x2^2)).*
        (30+(2*x1-3*x2)^2 .*
        (18-32*x1+12*x1^2+48*x2-36*x1.*x2+27*x2^2));
ML(p,i) eq1;
  METHOD = bfgs go nr ;
  MODE = minimum;
  GLOBOPT = 10000 100 .0001 4 ;
  MAXIT = 100;
```

This example demonstrates GO being used to minimize the Goldstein-Price function. ML is used on eq1 using `bfgs` as the initial step method, `go` the remaining steps, except for the final step (where one needs the Hessian) which is estimated using Newton-Raphson (`nr`). The feasible set is specified

using LOWERB and UPPERB in the PARAM statement.

See Also ML, NLS

Purpose

Creates a vector of log likelihoods for a Gompertz process.

Format

z = GOMPERTZ (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, index.
pvec	literal, positive parameter

Output

z	Vector of log likelihoods.

Remarks

The Gompertz distribution has been extensively used in the modeling of mortality data, and is suitable for modeling data with monotone hazard rates that change exponentially with time.

The Gompertz proportional hazards model is specified as:

$$H(t, x, \beta) = \exp(\gamma t) \exp indx$$

where $H(t, x, \beta)$ is the hazard function, and γ is a parameter that controls the shape of the baseline hazard.

indx is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The Gompertz model conventionally uses a linear index.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and the second column taking a value of unity if censored, else zero.

The baseline survival measures can be derived by setting the index to just the constant.

See the "General Notes for Non-Linear Models" under NLS. For residuals and survival measures, see the description under DURATION. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2 gama;
FRML eq0 indx = b0 + b1*arrtemp + b2*plant;


1  FRML ex1 llfn =  gompertz(fail, indx, gama);
   ML (p,i) eq0 ex1;

2  FRML ex2 llfn =  gompertz(fail~censor, indx, gama);
   ML (p,i) eq0 ex2;
   hr = exp(coeff);
   "Hazard Ratio " hr; call keyw;

3  CONST b1 b2;
     VALUE = 0 0;
   ML (p,i) eq0 ex1;
```

In example 1, a Gompertz model is estimated using maximum likelihood, with the index defined in eq0, and the log likelihood in eq1.

Example 2 shows a similar estimation when some of the data is censored. The hazard ratio is simply the exponent of the coefficients.

Example 3 shows how one would compute the constant only model.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Estimates the coefficients of a non-linear equation or system of equations using generalized method of moments.

Format

GMM (*options*) *elist ;*
BOUND = *level ;*
EQCON = *cnstrnt ;*
EQSUB = *macro ;*
INST = *instlist ;*
MAXIT = *maxit ;*
MAXSQZ = *maxsqz ;*
METHOD = *meth ;*
MODE = *mode ;*
POSDEF = *pdname ;*
SIMANN = *simann ;*
STEP = *step ;*
TITLE = *title ;*
TOL = *tolerance ;*
TRUST = *trust ;*
WEIGHT = *wtname ;*
WINDOW = *window ;*

Input

options	optional, print options.
elist	literal, required, equation list.
level	numeric, optional, percentage confidence level.
cnstrnt	literal, optional, list of constraint equations.
macro	literal, optional, macro equation list.
instlist	literal, required, list of instruments.
maxit	numeric, optional, maximum number of iterations (20).
maxsqz	numeric, optional, maximum number of squeezes (10).
meth	literal, optional, algorithm list (`NR NR NR`).
mode	literal, optional, estimation mode (`GMM`).
pdname	literal, optional, positive definite algorithm (`NG`).
simann	numeric, optional, SA options (`5 .85 100 20`).
step	literal, optional, step type (`LS`).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
trust	numeric, optional, TR options (`.1 1 .001 3`).

	wtname	literal, optional, weighting variable.
	window	literal/numeric, optional, spectral window.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
GRADVEC	Gradient vector.
LGCOEFF	Vector of Lagrangian coefficients.
QF	Quadratic form.
VCOV	Parameter covariance matrix.
COVU	Residual Covariance matrix.

Remarks

Generalized method of moments estimation (Hansen, 1982) requires the estimation of coefficients β using a set of instruments Z that satisfy the orthogonality conditions:

$$E(\epsilon(\beta)'Z) = 0$$

where ϵ are the (stacked) residuals:

$$\epsilon(\beta) = y - f(X,\beta)$$

and $E(\epsilon\epsilon') = \Omega$. A natural objective function - the minimum distance - is the quadratic form:

$$\text{QF}(\beta) = \epsilon(\beta)'ZW^{-1}Z'\epsilon(\beta)$$

where the optimal weighting matrix $W = Z'\Omega Z$. The estimated asymptotic covariance matrix for β is:

$$\text{Var}(\beta) = \left[G'Z(Z'\Omega Z)^{-1}Z'G\right]^{-1}$$

where G is $\partial\epsilon(\beta)/\partial\beta'$.

Each structural equation is specified as a Type II FRML, and the instruments (Z) are specified in an INST statement. The entire set of instruments is used for each equation. Estimation occurs in three stages. In the first, consistent (though inefficient) estimates of β are derived on the assumption that ϵ is homoscedastic with no autocorrelation; thus these are the 2SLS estimates.

Using the consistent estimate of Ω derived from this first stage process, a parameter estimate is derived; these are the 3SLS estimates. Using these parameter estimates, an efficient estimate of β can be derived taking into account heteroscedasticity and or serial correlation. For a system of equations, with the homoscedastic/no autocorrelation assumption, the GMM estimates coincide with 3SLS estimates. 2SLS and 3SLS estimates can be derived by specifying *mode* as 2SLS or 3SLS respectively.

Only the NR and SA step size methods are available with GMM (or non-linear 2SLS / 3SLS). The METHOD option is used exactly as in NLS. Robust estimation of the Ω matrix is specified using ROBUST as the final iteration method - this generates the White estimator for heteroscedastic disturbances. The Newey-West (1987) estimator for autocorrelated disturbances is implemented if ROBUST is specified and the spectral window and lag length is given in *window* – for details, see the WINDOW reference section

See the "General Notes for Non-Linear Models" under NLS, and the examples given in test02.prg.

Example

```
PARAM a0 a1 a2 b0 b1 b2 ;
FRML eq1 y1 = a0 + a1*x1 + a2*x2;
FRML eq2 y2 = b0 + b1*x3^b2;

1.  GMM (p,d,i) eq1 eq2;
       METHOD = nr nr robust;
       INST = c x1 x4 x5;

2.  GMM (p,i) eq2 ;
       METHOD = nr nr robust;
       MAXIT = 40;
       INST = c x1 x4 x5;
       WINDOW = 2;

3.  GMM (p,i) eq2 ;
       MODE = 2SLS;
       INST = c x1 x2;
```

In the first example, the system of equations eq1 and eq2 are estimated by GMM. The weighting matrix used at the second stage uses the White heteroscedastic estimator. Descriptive statistics (d) are displayed, and the coefficient values are printed at each iteration (i), with a pause (p) after each screen display.

The second example is a non-linear single equation GMM estimation, but the weighting matrix is the Newey-West heteroscedastic and autocorrelated consistent estimator, with a lag length of 2, and a BARTLETT (default) window.

The third example shows an alternative method of specifying a non-linear 2SLS estimation in GAUSSX.

See Also FRML, ML, NLS, TITLE, WEIGHT, WINDOW

References

Davidson R., and J.G. MacKinnon (1993), *Estimation and Inference in Econometrics*, Oxford University Press, Oxford.

Greene, W.H. (1993), *Econometric Analysis*, 2nd ed. Macmillan, New York.

Hansen, L.P. (1982), "Large Sample Properties of Generalized Method of Moments Estimation", *Econometrica*, Vol. 50, pp. 1029-1054.

Newey, W.K., and K.D. West (1987), "A Simple Positive Semi-Definite Heteroskedasticity and Autocorrelation Consistent Covariance Matrix", *Econometrica*, Vol. 55, pp. 703-708.

Purpose

Graphs one variable against another.

Format

GRAPH (*options*) *var1 var2* ;
FNAME = *filename ;*
GROUP = *grouplist ;*
MODE = *mode ;*
SYMBOL = *symlist ;*
TITLE = *title ;*

Input

options	optional, print options.
var1	literal, required, first variable.
var2	literal, required, second variable.
filename	literal, optional, macrofile.
grouplist	literal, optional, group variable list.
mode	literal, optional, graph mode (LINE).
symlist	literal, optional, symbol description.
title	string, optional, user defined title.

Remarks

The GRAPH command allows one variable to be graphed against another. The scale is automatically set. Graphing by groups is available using the GROUP option.

General Notes

GAUSSX supports two graphic packages. In the default, GRAPH uses GAUSS's Publication Quality Graphic (PQG) routines. The graph can be customized by specifying PQG global variables before the GRAPH command. The graphic file graphic.tkf is written on the SAMPLE path specified in the GAUSSX desktop.

The other package, GAUSSPlot, (if installed), will be used if specified in the GAUSSX configuration file, or by using the OPTION command. Under GAUSSPlot, the graph is most easily customized by creating a macro file of the changes made to the graph interactively, and specifying this file in *filename*. Customization can also be specified using *symlist*.

The default graph mode is a LINE graph. A scatter graph can be obtained using MODE = SCATTER. The default display mode - colour or mono can be explicitly set using COLOUR/MONO in the OPTION command.

Print options include *p* – pause until the graphic is closed, *m* – display for five seconds (PQG), *h* – print graph, and *r* – rotate graph (PQG). Examples of GRAPH are given in tutor.prg and in test53.prg.

Example

```
1.   GRAPH x1 x2;

2.   GRAPH x1 x2;
        MODE = SCATTER;

3.   OPTION pqg;
     _pbox = 1; _pcolor = 5;
     GRAPH (p) x1 x2;
        TITLE =  X1 vs. X2 ;

4.   OPTION gplot;
     GRAPH (p) x1 x2;
          ...
     GRAPH (p) x1 x2;
        FNAME = test4.mcr;
```

In the first two examples, observations in the current sample for each of the elements in x1 are graphed against the corresponding element of x2. The first example generates a line graph, while the second example creates a scatter graph.

In the third example, a graphic screen is displayed using PQG, and execution pauses (p) until the graph is closed. The line is colored green, and a box is drawn round the screen, and the user defined title is displayed.

Example 4 shows how a graphic can be customized using GAUSSPlot. From the GAUSSPlot window generated with the first GRAPH command, a macro record session is initiated by selecting files/macro/record from the GAUSSPlot menu. The macro is saved with the name test4 in the GAUSSX data path folder - typically gauss\dat. The graph is customized using the interactive GAUSSPlot GUI by double clicking the graphic element to be changed. The macro is saved when the stop macro button is clicked. Running the GRAPH

command with the macro specified in FNAME creates the customized graph.

See Also GROUP, OPTION, PLOT, TITLE

Purpose

Allows a GAUSSX command to be repeated for different groups.

Format

GAUSSX COMMAND *vlist ;*
GROUP = *grouplist ;*

Input

vlist literal, required, variable list.
grouplist literal, required, group variable list.

Remarks

It is often convenient to be able to specify a descriptive command to be replicated over a number of groups – this is the equivalent of the keyword BY in SAS. This could be done using multiple sample statements, but the GROUP option is simpler. For each replication, the data used is determined by both the current group and the existing sample. It can be used as an option for any descriptive command, as well as most estimation commands. A header is printed before each iteration stating the current values of the group variables.

An example is given in test01.prg.

Example

```
COVA x1 x2 x3;
   GROUP =  z1 z2;
```

In this example, a COVA is undertaken on every combination of z1 and z2. A header is printed giving the current values of these variables before each output. Thus if z1 takes five discrete values, and z2 takes three values, the COVA command will be replicated fifteen times.

See Also

COVA, GRAPH, NLS, OLS, PLOT, PRINT

Purpose
Creates a vector of log likelihoods for a Gumbel process.

Format
z = GUMBEL (*y*, *indx*, *pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, location index.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks
The Gumbel model can be used to estimate duration data. The expected value of loc_i is parameterized as:

$$E(loc_i) = indx_i$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

The Gumbel distribution is also known as the largest extreme value distribution. The expected value of location is the mode of *y*. The coefficients, β and *scale*, are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The scale parameter must be positive.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale; value = 1;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn =  gumbel(fail,indx,scale);
   ML (p,i) eq0 eq1;
     METHOD = nr bhhh bhhh;

2  FRML eq2 llfn =  gumbel(fail~censor,indx,scale);
   ML (p,i) eq0 eq2;
```

In example 1, a largest extreme value model is estimated using maximum likelihood, with the index defined in eq0, and the log likelihood in eq1. Example 2 shows the same estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Estimate the coefficients for the sample selection model using the Heckman two step estimation procedure.

Format

HECKIT (*options*) *elist ;*
CATNAME = *categories ;*
MAXIT = *maxit ;*
METHOD = *method ;*
TITLE = *title ;*
TOL = *tolerance ;*
WEIGHT = *wtname ;*

Input

options	optional, print options.
elist	literal, required, equation list.
categories	literal, optional, list of category names.
maxit	numeric, optional, maximum number of iterations (20).
method	literal, optional, covariance method (NONE).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
LLF	Log likelihood.
RSQ	R-squared.
RBARSQ	RBAR-squared.
VCOV	Parameter covariance matrix.

Remarks

The Heckman two step estimation procedure for the sample selection model has been implemented as a single command, with the option for the corrected (Greene) asymptotic parameter covariance matrix. *elist* consists of two elements - first the equation name for the PROBIT, followed by the equation name for the OLS. Sample selection, etc. is carried out automatically. The estimated Mills ratio is stored in the variable _LAMDA.

Print options include p —pause after each screen display, d —print descriptive statistics, e —print elasticities, i —print parameters at each iteration, q —quiet - no screen or printed output, s —print diagnostic statistics, and v —print parameter covariance matrix. These print options apply to both the PROBIT and the OLS estimation.

There are three available covariance methods for the OLS estimation. The default is NONE. Heteroscedastic-consistent variance-covariance matrix of parameters, corrected for the degrees of freedom, is available by setting *method* to ROBUST. Greene (1981) has proposed an asymptotic parameter covariance matrix, corrected both for heteroscedasticity and for the fact that Mills ratio is an estimated quantity; this can be specified by setting *method* to GREENE.

The variables described in "Outputs" are the outputs from the OLS estimation.

See the "General Notes for Linear Models" under OLS, QR, and the examples given in test08.prg.

Example

```
FRML eq1 sex c x1 x2 x3;
FRML eq2 wage c x1 z1 z2 z3;
HECKIT (i,p,d) eq1 eq2;
   METHOD = GREENE;
   CATNAME = male female;
```

In this example, a binomial probit is estimated - sex takes one of two values; the explanatory variables are x1, x2 and x3. The user can specify names for each category by using the CATNAME option. The sample is then set for those cases for which sex equals unity, and an OLS is carried out on eq2, with Mills ratio as an additional explanatory variable. The standard errors are based on the Greene corrected covariance matrix. The sample is reset to the

original sample at the end of the estimation.

See Also FRML, OLS, QR, TITLE, WEIGHT

References Greene, W.H. (1981), "Sample selection bias as a specification error: Comment", *Econometrica*, Vol. 49, pp. 505-513.

Heckman, J.J. (1979), "Sample selection bias as a specification error", *Econometrica*, Vol. 47, pp. 153-161.

Purpose

Creates a vector of log likelihoods for an integrated GARCH process.

Format

z = IGARCH (*resid, avec, bvec*);
z = IGARCH_T (*resid, avec, bvec, dvec*);

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for the ARCH process.
bvec	literal, vector of parameters for GARCH process.
dvec	literal, distributional parameter (ν).

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The structural coefficients and the coefficients of the IGARCH process are estimated using constrained maximum likelihood. The IGARCH model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ h_t &= \alpha_0 + \sum_{i=1} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1} \beta_j h_{t-j} \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The α are the vectors of the weights for the lagged ϵ^2 terms; this is the ARCH process. The β are the weights for the lagged h terms; this is the GARCH process.

avec is a vector of parameters giving the weights for the lagged squared residuals. The first element, which is required, gives the constant. *bvec* is the vector of parameters for the GARCH process. Thus this is a standard GARCH model, but with the identity restriction that:

$$\sum_{i=1} \alpha_i + \sum_{j=1} \beta_j = 1$$

Note the stationarity conditions described under GARCH.

See the "General Notes for GARCH" under GARCH, and the "General Notes for Non-Linear Models" under NLS.

Example

```
OLS y c x1 x2;
sigsq = serˆ2;
PARAM c0 c1 c2;
   VALUE = coeff;
PARAM a0 a1 a2 b1 ;
   VALUE =  sigsq .6 .2 .2 ;
FRML cs1 a0 >= .000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 b1 >= 0;
FRML cs5 a1+a2+b1 == 1;
FRML eq1 resid = y - (c0 + c1*x1 + c2*x2);
FRML eq2 lf = garch(resid,a0|a1|a2,b1);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4 cs5;
```

In this example, a linear IGARCH model is estimated using constrained maximum likelihood, with OLS starting values. The residuals are specified in eq1, and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive – in particular, since the constraint cs5 must hold, the initial values of the parameter should not violate this constraint.

Source GARCHX.SRC

See Also GARCH, EQCON, FRML, ML, NLS

Purpose Find the inverse values of a function.

Format ix = *INVERT (* &f, x0, z, kval *);*

Input

&f	Pointer to the function $f(x, z)$.
x0	literal, scalar or *N*x1 vector of starting value for *x*.
z	literal, scalar or *N*x1 vector, optional argument
kval	literal, scalar or *N*x1 vector, numeric, objective value.

Output

ix	value of x such that $f(ix, z) = kval$.

Remarks This procedure inverts a given function using Newton's method. Thus good starting values are essential. The function has two arguments, *x*, for which the inverse is required, an *z*, a second argument so as to provide flexibility. A missing value is returned if convergence is not achieved.

INVERT is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
proc sincos(x,z);
   retp(sin(x).*cos(x));
endp;
kvec = { .3, .4, .5 };
x0 = .2;
ix = invert(&sincos,x0,0,kvec);

 ix';    0.32175055   0.46364761   0.7853979

 sincos(ix,0)';   .3000   .4000   .5000
```

This example inverts the *sincos* function for the values shown in *kvec*.

Source GXPROCS.SRC

Purpose

Creates a vector of log likelihoods for an inverse Gaussian process.

Format

z = INVGAUSS (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, location index.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks

The inverse Gaussian model can be used to estimate duration data. The expected value of y_i is parameterized as:

$$E(y_i) = indx_i$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index and *pvec* are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn =  invgauss(fail, indx, scale);
   ML (p,i) eq0 eq1;

2  FRML eq2 llfn =  invgauss(fail~censor,indx, scale);
   ML (p,i) eq0 eq2;
```

In example 1, an inverse Gaussian model is estimated using maximum likelihood, with the index defined in eq0, and the log likelihood in eq1. Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Estimates the coefficients in a state space model, in which the coefficients follow a random process over time.

Format

KALMAN (*options*) *arglist ;*
OPLIST = *oplist ;*
VLIST = *vlist ;*

Input

options	optional, print options.
arglist	literal, required, variable list or equation name.
oplist	literal, optional, program options.
vlist	literal, optional, time varying coefficients.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
RSS	Residual sum of squares.
SER	Standard error of the regression.
LLF	Log likelihood.
VCOV	Parameter covariance matrix.

Remarks

The Kalman filter model allows for the vector of coefficients β in the classical linear model to randomly change over time. The model is specified in two parts - the measurement equation and the transition equation. The structural equation:

$$y_t = \alpha + X_t\beta_t + \epsilon_t \qquad \epsilon_t \sim N(0, H)$$

is standard, except that the kx1 vector β changes over time. Note that it is assumed that the model is homoscedastic. The transition equation specifies the time path of β:

$$\beta_t = \gamma + T\beta_{t-1} + \mu_t \qquad \mu_t \sim N(0, \sigma^2 Q)$$

Thus once an initial value of β_0 is chosen, then the solution for β at each time period (the state vector) will depend only on the matrix T, and the stochastic vector μ. Conditional on the dependent variable y and the independent variables X, the model can be evaluated once the system matrices are specified. These are:

α	a scalar constant in the measurement equation.
H	the variance of the residuals in the measurement equation (assumed homoscedastic).
γ	a *k*x1 vector of constants in the transition equation.
T	a *k*x*k* matrix of transition coefficients.
Q	a *k*x*k* symmetric matrix of the variance of the residuals in the transition matrix (up to a scalar factor).
β_0	a *k*x1 vector of coefficients at time zero in the structural equation.
Ω_0	a *k*x*k* symmetric matrix of the variance of β_0 (up to a scalar factor) .

The form of the structural equation is specified in the same manner as in OLS, using either a variable list, or a Type I FRML command. The data need not all fit in core. Weighting is not applicable for KALMAN. Print options include p — pause after each screen, d — print descriptive statistics, and q —quiet - no screen or printed output.

The program control options are specified in *oplist*. The options available are:

B0 =	vector of prior coefficients β_0 for the structural equation. P0 must also be specified. In the default, β_0 is estimated from the first *k* observations in the sample.
P0 =	covariance matrix (Ω_0) of β_0. σ^2 is factored out of this matrix. B0 must also be specified. In the default, this is set to $(X_k'X_k)^{-1}$ where X_k is the matrix of RHS variables for the first *k* observations of the sample.
CTRANS =	constant vector (γ) for the transition equation. In the default this is a *k*x1 vector of zeros.
BTRANS =	matrix of transition coefficients (T). In the default, this is a *k*x*k* identity matrix.
VTRANS =	covariance matrix (Q) of the residuals in the transition equation. σ^2 is factored out of this matrix. In the default this is a *k*x*k* identity matrix.
CMEAS =	constant scalar (α) for the measurement equation. In the default this is zero.
VMEAS =	constant, homoscedastic variance term (H) of the residuals in the measurement equation. Default is the identity matrix.

STATE = *svec*. The state vectors are stored using *svec* as the root. Thus if there are three coefficients in the measurement equation, then the state vectors are stored as *svec1*, *svec2* and *svec3* respectively.

PRDERR = vector of one step ahead prediction errors.

VARPRDER = vector of variance of the one step ahead prediction errors.

RECRES = vector of recursive residuals.. These can also be generated using the TEST command.

SMOOTH/[NOSMOOTH] Specifies whether the state vector is smoothed.

PRINT/[NOPRINT] Specifies whether a description of the various matrices actually used should be printed out. This is useful for debugging.

The matrices T, Q, H, α, γ, Ω_0, and the vector β_0 can be specified by the user, or default values are used. Optimally, they can be optimally chosen using the KALMAN process under ML, prior to running KALMAN. If they are specified by the user, then this must occur before the Kalman estimation, using standard GAUSS commands. For each matrix, each individual element may be a number, a parameter or constant name, or the name of a vector. If the element is a scalar, the value is fixed for the entire estimation. If it is a vector, which must be created beforehand, then at each time period during the Kalman estimation, the appropriate value of the vector is substituted into the element of the matrix. In this way, both the matrix of transition coefficients (T) and the covariance matrix of transition residuals (Q) can vary over time. The constant terms (α, γ) can also be time varying. The names of these vectors must be given in *vlist*.

The regression results reported by KALMAN are based on the residuals created at each time interval. The likelihood function is from Harvey (1981). The coefficients reported are the values of the evolving state vector at the last observation. Thus, if there are *n* observations, then the coefficient values are β_0 after having been transformed by the transition equation *n* times. If the default matrices are used, but Q is set to the null matrix, the Kalman process and recursive estimation are identical.

The computed state vectors, residuals and variances are stored in the GAUSSX workspace, and can be printed in the normal way. Note that the FORCST

statement used after a Kalman estimate will return fitted values based on the last value of the state vector - the coefficients as printed out by the Kalman procedure.

A number of examples of the KALMAN procedure are given in test15.prg.

Example

```
1.   KALMAN m c r gnp;
        OPLIST =  recres=resid;

2.   let tmat[2,2] = t11 t12 t21 t22;
     KALMAN y c x;
        OPLIST = btrans=tmat state=kfit ctrans=.5 smooth;
     PRINT (p) kfit1 kfit2;

3.   let tmat[3,3] =   1  0 0
                       0 a2 0
                       0  0 t33;
     GENR trend = numdate(_ID);
     GENR t33 = .8 + .01*trend;
     KALMAN (p,d) y c x1 x2;
        OPLIST = btrans=tmat print;
        VLIST  = t33;
```

In the first example, a Kalman filter is applied to the regression of `m` on `c`, `r`, and `gnp`. Default values are used for the matrices - thus T and Q are identity matrices, and β_0 and Ω_0 are estimated from the first three observations. The recursive residuals are stored in a vector called `resid`.

In the second example, a Kalman filter is applied to the regression of `y` on `c` and `x`. A transition matrix `tmat` is specified, where the elements have already been defined. A constant `.5` is specified in the transition equation. The filter produces smoothed estimates of the state vectors, corresponding to the coefficients on `c` and `x`; these are stored, and subsequently printed, as `kfit1` and `kfit2`.

The third example shows the transition coefficient matrix T as varying over time. In this example, T is specified as a diagonal matrix with the first el-

ement on the principal diagonal being unity, the second element being a2, (a parameter or constant previously estimated), and the third element, t33, changing over time. The form of t33 is specified in a GENR statement prior to the Kalman estimation. In the estimation, BTRANS is set equal to tmat in the OPLIST option. Since tmat has time varying components, they must be described in VLIST. A description of the various matrices is requested by the PRINT option in OPLIST.

See Also KALMAN, OLS, TEST

References

Kalman, R. (1960), "A New Approach to Linear Filtering and Prediction Problems", *Journal of Basic Engineering, Transactions ASME, Series D*, Vol. 82, pp. 35-45.

Harvey, A,C. (1981), *Time Series Models*, Philip Allen, London.

Judge, G.G., *et. al.* (1985), *The Theory and Practice of Econometrics*, 2nd edition, Wiley, New York.

Purpose — Creates a vector of log likelihoods from a Kalman filter process.

Format — *z* = KALMAN (*y, x*);

Input

y	literal, dependent variable.
x	literal, matrix of independent variables.

Output

z	Vector of log likelihoods.

Remarks — The KALMAN command can be used within a FRML command. It is used within the context of maximum likelihood to estimate the parameters of the various Kalman matrices, and requires that all the observations can reside in core. In this context, the command does not produce a vector of state matrices, but only returns the log likelihood. The options specified in the OPLIST command for KALMAN also apply in this context.

See the "General Notes for Non-Linear Models" under NLS. Also see KALMAN for a full discussion of the Kalman filter, and the options available under GAUSSX . An example of a Kalman estimation procedure running under maximum likelihood is given in test15.prg.

Example

1.
```
PARAM a1 a2 a3;
   VALUE = 1 1 1;
let tmat[3,3] =  a1  0 0
                  0 a2 0
                  0  0 a3;
FRML eq1 plst = a1 + a2 + a3;
FRML eq2 lf = kalman(y,c~x1~x2);
ML (p,i) eq1 eq2 ;
   OPLIST = btrans=tmat print;
KALMAN (p) y c x1 x2;
   OPLIST = btrans=tmat;
```

2.
```
PARAM a1 a2 a3;
   VALUE = 1 1 1;
let tmat[3,3] =  a1  0 0
                  0 a2 0
```

```
                    0  0 t33;
FRML eq1 t33 = a3 + .005*r;
FRML eq2 plst = a1 + a2 + a3;
FRML eq3 lf = kalman(y,c~x1~x2);
ML (p,i) eq1 eq2 eq3;
   OPLIST = btrans=tmat print;
   VLIST  = t33;
```

These two examples show how the parameters of a Kalman filter coefficient transition matrix could be estimated using ML. In both these examples it is assumed that both the transition matrix (T) and the covariance matrix (Q) are diagonal. The default – an identity matrix – is used for Q, and the initial values for β and Ω are based on the first three observations. In the first example, the T matrix, which is time invariant, is defined using a GAUSS statement, with parameter values for the elements on the principal diagonal. The first equation `eq1` is a dummy equation - it is only required since GAUSSX scans the equations to see which parameters it needs to estimate. The log likelihood is returned from the second equation `eq2`, which defines the structural model - a LHS variable (`y`), and three explanatory variables (`c`, `x1`, and `x2`) written as a matrix. `tmat` is specified as the coefficient transition matrix in the OPLIST statement. The ML estimates of the elements of `tmat` are returned, and stored in `a1`, `a2` and `a3`. The last KALMAN command reports the standard Kalman results, based on these parameter values.

The second example shows how a time varying matrix can be estimated. The third coefficient, `a3` is now replaced with a vector, `t33`, which is specified in `eq1` as a linear function of `r`, with the intercept to be determined as a parameter. The second and third equations are unchanged. The time varying vector is specified on the VLIST option.

Source KALMAN.SRC

See Also KALMAN, ML, PARAM

Purpose To retain only the specified variables in the current GAUSSX work-space.

Format KEEP *vlist ;*

Input *vlist* literal, optional, variable list.

Remarks The KEEP statement retains only the specified variables in the GAUSSX work-space. The vectors C, _ID and SAMPLE are also retained. The current SMPL remains in effect. Vectors with missing values are saved as is.

Example

```
KEEP x1 x2 x3;
```

A GAUSSX workspace is created with the variables x1, x2 and x3.

See Also DROP, RENAME, STORE

Purpose To create a vector of lagged values.

Format *z* = LAG (*x*, *n*);

Input
x literal, required, variable name.
n numeric, required, lag length.

Output
z Lagged vector.

Remarks The LAG command can be used within a GENR command. It creates a lagged variable, with the length of the lag specified by *n*, over the current sample. The initial *n* observations, as well as observations outside the sample, are set to missing value. The default maximum lag is 12; this can be changed using the OPTION command.

The LAG command can also be used within a non-linear FRML providing that the data can all reside in core. If this is not possible, a lagged variable can be created using a GENR, and the new variable can be used instead. LAG is also used for dynamic solutions of equations - see the example in SOLVE.

Example

```
1.   GENR y = x1 + .5*lag(x1,1) + .25*lag(x1,2);

2.   GENR plag = lag(p,2);
     FRML eq1 q = a0 + a1*plag;
     FRML eq2 q = a0 + a1*lag(p,2);
```

In the first example, a vector y is created from the weighted sum of x1, x1 lagged once and x1 lagged twice. In the second example, the two equations are identical; however the second would be required for a dynamic SOLVE.

See Also GENR, PDL, SOLVE

Purpose

Draws a Latin Hypercube Sample from a set of uniform distributions for use in creating a Latin Hypercube Design.

Format

p = LHS (*n, k, dsgn*);

Input

n	scalar, number of runs.
k	scalar, number of factors.
dsgn	scalar or matrix, design parameters.

Output

p	*n*x*k* matrix of probabilities.

Remarks

Latin hypercube sampling (LHS) was developed to generate a distribution of collections of parameter values from a multidimensional distribution. A square grid containing possible sample points is a Latin square iff there is only one sample in each row and each column. A Latin hypercube is the generalization of this concept to an arbitrary number of dimensions. When sampling a function of k variables, the range of each variable is divided into *n* equally probable intervals. *n* sample points are then drawn such that a Latin Hypercube is created. Latin Hypercube sampling generates more efficient estimates of desired parameters than simple Monte Carlo sampling.

This program generates a Latin Hypercube Sample by creating random permutations of the first *n* integers in each of *k* columns and then transforming those integers into *n* sections of a standard uniform distribution. Random values are then sampled from within each of the *n* sections. This sampling scheme does not require more samples for more dimensions (variables); this independence is one of the main advantages of this sampling scheme. Another advantage is that random samples can be taken one at a time, remembering which samples were taken so far.

Once the sample is generated, the uniform sample from a column can be transformed to any distribution by using the quantile functions, e.g. `normal_cdfi`. Different columns can have different distributions.

There are a number of algorithms available; the algorithm selected is determined by the value of *dsgn*:

`0`	Standard LHS.

1	Nearly orthogonal LHS, no criteria.
Ω	Correlated LHS, no criteria, where Ω is a *k*x*k* symmetric covariance matrix.
v	Nearly orthogonal LHS, with criteria, where *v* is a 3x1 design vector.

The specification of *v* is given below:

v[1]	Fill: 0 - even spread, 1 - end to end.
v[2]	Maximum number of column exchanges per factor.
v[3]	Criteria: 1. Average absolute correlation 2. Condition number 3. Maximum VIF - main effect 4. Maximum VIF - main effect + quadratic; 5. Maximum VIF - main effect + cross; 6. Maximum VIF - main effect + quadratic + cross; 7. J2 optimality 8. Modified L2 (Cioppa) 9. Euclidean minmax

If *v* is specified, the number of levels $m = n/k$, must be integer. The effect of the `fill` parameter can be seen as follows:

k=1, m = 4, fill = 0: p = .125 .375 .625 .875
k=1, m = 4, fill = 1: p = 0 .3333 .6666 1

LHS is pure GAUSS code, and can be used independently of GAUSSX. An example of LHS is given in test62.prg.

Example

```
library gaussx ;
rndseed 12345;
n = 30; k = 6;
fill = 0; ntry = 1000; crit = 2;
dsgn = fill | ntry | crit;
```

```
p = lhs(n,k,dsgn);
x = normal_cdfi(p,0,1);
```

In this example, a 30x6 nearly orthogonal Latin Hypercube Sample is derived using the best condition number as the criteria. This creates a 30x6 matrix of probabilities, which are then used to create a set of standard normally distributed variates, each column being orthogonal to every other column..

Source LHS.SRC

See Also COPULA, CORR, MVRND

Purpose Preprocessor command to reinstate the command file listing.

Format #LIST ;

Remarks Normally the entire GAUSSX command file listing is provided in the output file, prior to the execution listing. The command file listing can be selectively suppressed by using the #LIST and #NOLIST commands. #NOLIST; switches off the listing. #LIST; switches it back on.

Example

```
#LIST;
```

See Also #NOLIST, PAGE

Purpose

Assigns a single name to a list of GAUSSX variable names.

Format

LIST *listname vlist ;*
RANGE = *range ;*
SYMBOL = *rootname ;*

Input

listname	literal, required, list name.
vlist	literal, required, variable list.
range	numeric, optional, list range.
rootname	literal, optional, element name.

Remarks

It is frequently convenient to replace a long list of variable names with *listname*, such that *listname* can be used later in the program whenever the list of variables would have been used. *listname* may be any legal GAUSS variable name - and is thus limited to 8 characters.

vlist may include lags, and can also include the names of other lists; indeed, lists may be nested indefinitely. Since LIST is an executable command, the contents of a list can be redefined by maintaining the same listname, but changing *vlist*. In the case of nesting, list names that occur in the variable list will contain the variables as they exist at the time of execution.

The creation of a list of variable names with a common stem is also possible. The stem is specified in *rootname*, and the range is specified in *range*.

The LIST statement registers *listname* as a list, and places vlist into *listname*. Since *listname* is a global, it can be manipulated using standard GAUSS - see example 2.

Example

1.
```
LIST rhslist c gnp relp;
LIST ivlist c gnp inv rate;
FRML eq1 cons rhslist;
OLS cons c gnp relp;
OLS cons rhslist;
OLS eq1;
2SLS eq1;
   INST = ivlist;
```

```
2.  LIST avlist dum;
    avlist = 0 $+  AV  $+ ftocv(seqa(1,1,20),2,0);

3.  LIST xlist ;
       SYMBOL = xx;
       RANGE = 1 20;
```

The first example shows three OLS regressions that are exactly equivalent. The 2SLS code shows how a list can be used for the instruments. The second example shows how a list of variables AV01 to AV20 can be defined. In the third example, the list xlist consists of the twenty variable names xx1 through xx20.

Purpose To load data into GAUSSX

Format LOAD *vname* = *data ;*

Input
vname literal, required, variable name.
data numeric, required, data values.

Remarks The LOAD command allows data to be entered directly from the command file. The number of observations specified in data must be consistent with the current sample.

If a vector has previously been defined over a sample space that is longer than the current sample space, then the excluded observations are set to missing value if REPL has been set by the OPTION command (default), and remain unaffected if the NOREPL option has been set.

Example

```
LOAD gnp = 100 110 120 132 143 165 180 190 200;
```

In this example, nine values are loaded into a vector called gnp.

See Also GENR, OPTION

Purpose To load and compile gradient and Hessian procedures.

Format
```
LOADPROC ;
    GRADIENT = &gradname ;
    HESSIAN = &hessname ;
```

Input
&gradname string, optional, procedure name.
&hessname string, optional, procedure name.

Remarks The LOADPROC statement loads the gradient and Hessian procedures previously stored with the SAVEPROC command from the GAUSSX `data path`, and compiles them. This is much faster than recreating them using `symgrad` and `symhess`. Note that the model structure and parameter space must remain unchanged for the gradient and Hessian procedures to be valid.

A dummy procedure with the same name has to be specified for compilation integrity.

LOADPROC requires GAUSS 4.0 or higher.

Example
```
proc garchp; endp;

LOADPROC;
   GRADIENT =  garchp ;
ML (p,i,s) eq1 eq2;
   METHOD = nr bhhh nr;
   GRADIENT = &garchp;
```

This example retrieves the previously stored procedure `garchp`, and uses it for the analytic gradient in the ML estimation.

See Also ML, SAVEPROC

Purpose Creates a vector of log likelihoods for a logistic process.

Format *z* = LOGISTIC (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, location index.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks The expected value of y_i is parameterized as:

$$E(y_i) = indx_i$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index and *pvec* are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The scale parameter must be positive.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale; value = 1;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn = logistic(fail,indx,scale);
   ML (p,i) eq0 eq1;
     METHOD = nr bhhh bhhh;

2  FRML eq2 llfn = logistic(fail~censor,indx,scale);
    ML (p,i) eq0 eq2;
```

In example 1, a logistic model is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a logistic model estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose — Creates a vector of log likelihoods for a binomial logit model.

Format — *z* = LOGIT (*y, x*);

Input

y	literal, vector of alternative chosen.
x	literal, vector, utility.

Output

z	Vector of log likelihoods.

Remarks — The structural coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

Example

```
FRML eq1 xb = a0 + ln(a1*x1 + a2*x2);
FRML ellf llf = logit(y1,xb);
ML (p,i) eq1 ellf;
   METHOD = bhhh bhhh nr;
```

This example estimates a non-linear binomial logit model.

Source — PROBITX.SRC

See Also — ML, MNL, NLS, QR

Purpose
Creates a vector of log likelihoods for a loglogistic process.

Format
z = LOGLOG (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, index of the means.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks
The expected value of y_i is parameterized as:

$$E(\ln(y_i)) = indx_i$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index and *pvec* are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The scale parameter, *pvec* must be positive.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale; value = 1;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn = loglog(fail,indx,scale);
   ML (p,i) eq0 eq1;
     METHOD = nr bhhh bhhh;

2  FRML eq2 llfn = loglog(fail~censor,indx,scale);
    ML (p,i) eq0 eq2;
```

In example 1, a loglogistic model is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a loglogistic model estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose — Creates a vector of log likelihoods for a lognormal process.

Format — *z* = LOGNORM (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, location index.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks — The lognormal model can be used to estimate duration data. The expected value of y_i is parameterized as:

$$E(\ln(y_i)) = indx_i$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index and *pvec* are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The scale parameter must be positive.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale; value = 1;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;
FRML ec1 scale >= 0;

1  FRML eq1 llfn =  lognorm(fail,indx,scale);
   ML (p,i) eq0 eq1;
      eqcon = ec1;

2  FRML eq2 llfn =  lognorm(fail~censor,indx,scale);
   ML (p,i) eq0 eq2;
      eqcon = ec1;
```

In example 1, a lognormal model is estimated using constrained maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Repeat a block of code for each sector in a multisectored body of data.

Format

LOOP *symbol seclist ;*
.
GAUSS and/or GAUSSX code
.
ENDLOOP*;*

Input

symbol literal, required, symbol for sector component.
seclist literal, required, vector list of sector symbols.

Remarks

The names of the series must all have a generic part, common for the series across sectors, and a sector part that is common across series. Any number of GAUSS or GAUSSX statements may appear between the LOOP and ENDLOOP statements. GAUSSX substitutes the sector names for the sector symbol throughout; one must ensure that the resulting variable name does not exceed 8 characters.

LOOP..ENDLOOP is functionally equivalent to DOT..ENDDOT in TSP.

Example

```
LOOP # uk us can;
   FRML eq# ue# c trend infl#
   OLS (p,d) eq#
ENDLOOP;
```

In this example, a regression of unemployment (UE) against inflation (INFL) and trend is undertaken for the UK, US, and Canada. The first argument in the LOOP statement is the symbol that is used to represent the sector component in the series. The remaining arguments are the sector symbols. In the above example, the OLS is run three times, first for the UK, then for the US, then for Canada.

Purpose

Solves the linear programming problem.

Format

LP (*options*) *ename ;*
EQCON = *cnstrnt ;*
EQSUB = *macrolist ;*
MODE = *lpmode ;*
TITLE = *title ;*

Input

options	optional, print options.
ename	literal, required, objective function equation.
cnstrnt	literal, required, list of constraint equations.
macrolist	literal, optional, macro equation list.
lpmode	literal, optional, optimization mode ([maximize]/minimize).

Output

COEFF	Vector of coefficients.
LGCOEFF	Vector of Lagrangian coefficients.
LLF	Value at optimum.

Remarks

The LP command solves the standard linear programming problem - that is maximizing (or minimizing) a linear objective function subject to linear constraints and upper and/or lower bounds. The linear objective function is specified in *ename*, while the constraints are defined in the equations specified in *cnstrnt*. The coefficients are specified in a PARAM statement, along with the lower and upper bounds. In the default, LP maximizes the objective function; minimization occurs when *lpmode* is specified as minimum.

Print options include d – descriptive statistics, (if variables are included), p – pause after each screen display, and q – quiet - no screen or printed output.

An example is given in test45.prg

Example

```
PARAM x1 x2 x3 x4;
    LOWERB = 0 0 0 0;
    UPPERB = 100 100 100 100;
FRML eq1 cost =  2.0*x1  + 2.5*x2 + 1.8*x3 + 1.4*x4;
FRML ec1 x1 + x2 <= 200;
FRML ec2 x3 + x4 <= 400;
FRML ec3 x1 + x3 >= 325;
FRML ec4 x2 + x4 >= 300;
LP (p)  eq1;
  EQCON = ec1 ec2 ec3 ec4;
  MODE = minimize;
```

In this example, there are four parameters to be estimated (`x1, x2, x3, x4`) such that the objective function (cost) is minimized, subject to the four constraints. The parameters are specified in the `PARAM` statement, along with the upper and lower bounds. The linear objective function is specified in `eq1` while the constraints are specified in `ec1, ec2, ec3, ec4`. A minimization is carried out, since `mode` is specified as `minimize`.

See Also

FRML, PARAM

Purpose

Computes the maximum Lyapunov exponent for a time series.

Format

```
LYAPUNOV (options) vname ;
    MAXSTEP = gridstep ;
    OPLIST = oplist ;
    ORDER  = order ;
    PERIODS = periods ;
    RANGE = range ;
    TITTLE = title ;
```

Input

options	optional, print options.
vname	literal, required, variable name.
gridstep	literal, optional, grid resolution (20)
oplist	literal, optional, program options
order	literal, required, embedding dimension
periods	literal, optional, lags. (1 1)
range	literal, optional, replacement range.

Output

_LYAEXP	Lyapunov exponent.

Remarks

The Lyapunov exponent of a dynamical system is a measure that determines for a point of phase space how quickly trajectories that begin at this point diverge over time. The number of Lyapunov exponents is equal to the number of dimensions of the embedding phase space; however the maximal Lyapunov exponent (MLE) is usually reported, since it determines the predictability of a dynamical system.

The Lyapunov exponents L_i are calculated as

$$L_i = \lim_{t \to \infty} 1/t \log_2(r(0)/r_i(t))$$

which can be thought of as following the motion of an infinitesimally small sphere, with an initial radius $r(0)$, that starts from the point for which the exponent should be calculated. On its trajectory, it will expand unevenly, so that it becomes an ellipsoid with time-dependent radii $r_i(t)$ in each principal direction. A stable point has an MLE that is negative, a limit cycle has an MLE

that is zero, while a strange attractor (chaos) has an MLE that is positive. Thus a positive MLE implies that nearby points, no matter how close, will diverge to any arbitrary separation. Examples include brownian motion, as well as strange attractors.

GAUSSX uses the Wolf algorithm to evaluate the Lyapunov exponent. The embedded dimension (*order*) must be specified - it should be at least as large as the minimum number of dynamical variables needed to model the dynamics of the system. The lag used to reconstruct a phase space from a time series is specified as the first element of *periods*, and the evolution length is specified as the second element of *periods*. A rule of thumb is that the evolution length can equal the phase lag - however, small evolution lengths require larger processing time. The grid resolution is specified in *gridstep*, and the minimum and maximum separations are given in *range*

The program control options are specified in *oplist*, with default values in parentheses. The options available are:

DT =	Specifies the sampling rate for continuous functions - see the Lorentz example in test49.prg. For discrete functions, this usually takes the default value (1).
MAXBOX =	Specifies the maximum number of boxes allocated for the Wolf algorithm. While the default is usually sufficient, this value can be changed if a larger number is required (6000).
THMAX =	Specifies the maximum orientation error, in degrees (30).

Print options include p —pause after each screen display, and q —quiet - no screen or printed output. Additional information is available through the on-line help (Alt-H). An example is given in test49.prg.

Example

```
LYAPUNOV (p) x;
   ORDER = 3;
   PERIODS = 1 4;
```

In this example, the Lyapunov exponent of a time series (x) is investigated, using an embedded dimension of 3, a default time lag of 1, and an evolution size of 4 periods.

See Also CORDIM

References Wolf, A., J.B. Swift, H.L. Swinney, and J.A. Vastano (1985), "Determining Lyapunov Exponents from a Time Series", *Physica* Vol. 16D (3), pp 285-317.

Purpose

Given three of the following: principal, interest rate, term, periodic payment; evaluate the fourth.

Format

y = MCALC (*p, r, pmnt, n*);

Input

p	*N*x1 vector or scalar, loan amount.
r	*N*x1 vector or scalar, interest rate per period.
pmnt	*N*x1 vector or scalar, periodic payment.
n	*N*x1 vector or scalar, number of periods.

Output

y	*N*x1 vector or scalar, corresponding to the argument that is scalar zero.

Remarks

MCALC will return a vector or scalar corresponding to whichever argument is zero; it thus functions like a mortgage calculator. The interest rate is per period; thus an annual rate of 9% paid monthly for 20 years would have $r = .09/12 = 0.0075$, and $n = 12 * 20 = 240$.

MCALC is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
p = 100000;
r = .1;
let nper = 10 15 20 25;
pmnt = mcalc(p,r/12,0,12*nper);

pmnt' = 1321.5075  1074.6051  965.0217  908.7008
```

This calculates the monthly payments on a $100,000 mortgage at 10% amortized over 10, 15, 20 and 25 years

Source

FINANCE.SRC

See Also

AMORT, FV, PV

Purpose

Bayesian estimation using Markov Chain Monte Carlo simulation.

Format

MCMC (*options*) *vlist ;*
USERPROC = *&procname ;*
CATNAME = *catname ;*
PRIOR = *prior ;*
REPLIC = *replic ;*
TITLE = *title ;*
VALUE = *value ;*

Input

options	optional, print options.
vlist	literal, required, vector list or equation name.
&procname	literal, required, pointer to user proc.
catname	literal, required, label for each parameter.
prior	literal, optional, priors.
replic	numeric, replication info (1000 0 100).
title	string, optional, title.
value	literal, optional, starting values.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.

Remarks

The MCMC command carries out Markov Chain Monte Carlo simulation over a user defined procedure *procname*. The data is specified in *vlist*, which can be specified by using a list of variables, or by using an equation name(s) which has been previously specified as a Type I FRML command. Priors are specified in *prior*, these can be numeric values or variable names (scalar, vector or matrix). The names of each parameter is given in *catname*, which must the same length as the parameter vector that is updated in *procname*. An initial parameter value can be specified in *value*; the default is missing. At the end of the set of iterations, descriptive statistics are given for the elements given in *catname*.

Print options include i – show iteration count, p – pause after each screen display, q – quiet - no screen or printed output, and s – print convergence diagnostics, cumulants and confidence bands.

Replication information is specified in *replic*, which can consist of up to three elements. The first element is the total number of replications. The second element is the number of replications to be used for burn-in; the realizations created during the burn-in are not included in the statistics. The third element is the frequency of the printing of the iterations.

procname is a user defined procedure, which creates a realization of a continuous state Markov chain process. The type of realization is up to the user - the literature includes a Gibbs sampling algorithm, a metropolis chain, or the Hastings algorithm. *procname* takes no inputs, and returns no outputs. Rather, there are a number of helper utilities that do the basic input and update work:

`{y,x,k} = dataget;` Retrieve data specified in *vlist*; the first element of vlist becomes `y`, and the remainder becomes `x`. `k` is a vector of the number of columns in `x` for each equation.

`coeff = coeffget;` Retrieve starting values of coefficients from *value*. If no starting values are specified, `coeff` is specified as missing.

`rslt = priorget(vname,defval);` Retrieve parameter value from *prior* with name `vname`; if `vname` does not exist, the value specified in `defval` is used.

`call mcupdate(iter,coeff);` Updates the storage matrix with the current value of the coefficients (`coeff`), as well as printing out iteration (`iter`) information.

A number of examples of the MCMC procs are given in bayes.src in the gauss\src directory; these include:

AR(k) with heteroscedastic residuals.
binomial probit.
heteroscedastic binomial probit.
multinomial probit.

OLS with residuals distributed normal.
OLS with residuals distributed t.
OLS with heteroscedastic residuals.
Poisson.
Tobit.
SURE.

Convergence diagnostics are included with the s option. These include Geweke's NSE (Numerical Standard Error), RNE (Relative Numerical Efficiency), and a Chi-squared test for parameter stability. Additional information is available through the on-line help (Alt-H).

When adding your own MCMC code, use the code examples in bayes.src as a template, and add the proc name to the gaussx.lcg library file. A menu of the test applications is given in test29.prg; the examples are in the gauss\prg\mcmc directory.

Example

```
rndseed 123456;
FRML eq1 y c x1 x2;
MCMC (i,s) eq1;
   CATNAME  = const x1 x2 sig2;
   REPLIC   = 1100 100 200;
   USERPROC = &g_ols_t;
   PRIOR    = df=3;
```

In this example, an MCMC simulation is executed 1100 times over the function g_ols_t. Diffused priors are used, except for the degrees of freedom for the t distribution, which is set at 3. The endogenous variable is y, and the explanatory variables are a constant, x1 and x2. The coefficient vector updated by g_ols_t consists of the structural parameters c x1 x2 and the residual variance sig2. The first 100 realizations are scrapped as burn in. The iteration count is displayed every 200 iterations. When the realizations are complete, simulation statistics (mean, variance, quartiles) are displayed.

References

Chib, S. and Greenberg, E. (1996), "Markov Chain Monte Carlo Simulation Methods in Econometrics", *Econometric Theory*, 12, 409-431

Geweke, J. (1992), “Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments”, in J.O. Berger, J.M. Bernardo, A.P. Dawid, and A.F.M. Smith (eds.), *Bayesian Statistics 4*, 169-194. Oxford: Oxford University Press, 1992.

Gelman A., Carlin J., Stern H., and D. Rubin, (1995), *Bayesian Data Analysis*, CRC.

Gilks L., W. Richardson, and D. Spiegelhalter, (1996), *Markov Chain Monte Carlo in Practice*, CRC.

Robert, C.P, and G. Casella (2000), *Monte Carlo Statistical Methods*, Springer Verlag, New York.

Purpose

Undertakes a Monte Carlo Simulation over a block of code.

Format

MCS (*options*) BEGIN *;*
    CATNAME = *catname ;*
    REPLIC = *replic ;*
    OPLIST = *progopts ;*
    TITLE = *title ;*
    VLIST = *vlist ;*
    .
    GAUSS and/or GAUSSX code
    .
MCS END *;*

Input

options	optional, print options.
catname	literal, required, label for each parameter.
replic	numeric, optional, replication info (100 0 1).
progopts	literal, optional, program options.
title	string, optional, title.
vlist	literal, required, vector list or vector name.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
_MCITER	Current iteration number.
_MCMAT	MCS work matrix.

Remarks

The MCS command carries out a Monte Carlo simulation over a block of code, which can consist of GAUSS and/or GAUSSX commands. At the end of each replication, the elements in the vector vlist are extracted; vlist may be a vector list, or the name of an array. The length of *catname* must equal the number of elements pointed to in *vlist*. At the end of the simulation, descriptive statistics are given for the elements of *vlist*.

Replication information is specified in *replic*, which can consist of up to three elements. The first element is the total number of replications. The second

element is the number of replications to be used for burn-in; the realizations created during the burn-in are not included in the statistics. The third element is the frequency of the printing of the iterations. For jackknife, the total number of replications is the sample size, and burn-in are ignored.

During the simulation, a screen is presented which shows the current replication, the value of the parameters in the current replication, the number of warnings, and the time to completion. Some options can be changed at run time while the simulation is progressing by pressing any key, and then selecting the required option from the menu. The current iteration is stored in __mciter

Print options include i – print vlist after each iteration, p – pause after each screen display, q – quiet - no screen or printed output, and s – print cumulants and confidence bands.

The program control options are specified in *progopts*. The options available are:

SCREEN/[NOSCREEN]	Turns screen on/off.
OUTPUT/[NOOUTPUT]	Turns output on/off.
[WARN]/NOWARN	Warnings are enabled/disable.
[IGNORE]/EXCLUDE/ABORT	Specifies the response to a warning.
[SIMULATE]/BOOTSTRP/JACKNIFE	Specifies the type of simulation.

See the OPTION command for details on the first three choices. If a GAUSSX warning occurs during the simulation, one can choose to ignore it, exclude the case, or abort the simulation. If EXCLUDE is specified, the replication is repeated.

Three types of simulation can be carried out. The first, SIMULATE, is a standard Monte Carlo simulation. The BOOTSTRP option is identical, (the user chooses how the stochastic component is generated), and only affects the title. The JACKNIFE option results in *replic* being set to the sample size, and the MCS variances being scaled by the sample size. Examples of all three methods are given in test17.prg.

If there are *k* elements in the vector specified in vlist, and there are *n* replications, then the *n*x*k* matrix is stored in _mcmat, which is available after the Monte Carlo run.

Example

```
rndseed 123456;
MCS (i,s) BEGIN;
    CATNAME = val_g0 val_g1 val_g2 ser_g0 ser_g1 ser_g2;
    VLIST = cofstd;
    REPLIC = 50;
    OPLIST = nowarn exclude;
  GENR y = (2 + 3*x1)*x2^.5 + rndn(n,1);
  PARAM g0 g1 g2;
  FRML eq1 y = (g0 + g1*x1)*x2^g2;
  NLS eq1;
  cofstd = coeff|stderr;
MCS END:
```

In this example, a Monte Carlo simulation of a non-linear estimation is carried out. The parameters of interest are the coefficient values and their standard errors – they are stored in a vector called cofstd at the end of each replication. The MCS BEGIN command at the beginning of the block is informed of this vector in the VLIST statement, and a set of labels are given in CATNAME. The block of code is delineated with the MCS END command. After 50 replications, the simulation statistics will be produced.

References

Judge, G.G., *et. al.* (1988), *Introduction to the Theory and Practice of Econometrics*, 2nd ed. John Wiley & Sons, New York.

Purpose

Evaluates the probabilities for a maximum entropy class of problem.

Format

p = ME (*k*, *&fct*);

Input

k	scalar, required, number of alternatives.
&fct	literal, required, pointer to constraints procedure.
_meStepTol	global scalar, step size tolerance (default = 1e-8).
_mePrint	global scalar, output flag: 0 - off, 1 - on. (default=1)

Output

p *k*x1 vector of probabilities.

Remarks

The maximum entropy problem is solved by explicitly solving the first order binding conditions, The probabilities are constrained to be non-negative, as well as satisfying the adding up normalization constraint, and the moment consistency constraints specified in `&fct`. This is a pointer to a procedure that computes the moment consistency constraints. The procedure must have one input argument, the *k*x1 vector of probabilities, and one output argument, the *r*x1 vector of computed constraints that are to be equal to zero.

An example of ME is given in test26.prg.

ME is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
k = 6;  x = seqa(1,1,k);  y = 3.0;
proc mcproc(p);  retp( y-x'p);  endp;
p = me(k,&mcproc);
```

This example evaluates the probability of each fall of a biased dice based on the single observed “average” mean - in this case 3.0. For a fair dice, y would be 3.5.

Source

MAXENTX.SRC

References

Golan, A., G. Judge and D. Miller, *et. al.* (1996), *Maximum Entropy Econometrics: Robust Estimation with Limited Data*, John Wiley & Sons, New York.

Purpose Creates a vector of log likelihoods for a multivariate GARCH process.

Format *z* = MGARCH (*resid, x, c0, c1, amat, gmat*);

Input

resid	literal, matrix of residuals.
x	literal, matrix of weakly exogenous variables.
c0	literal, matrix of parameters for constants.
c1	literal, matrix of parameters for *x*.
amat	literal, matrix of parameters for ARCH process.
gmat	literal, matrix of parameters for GARCH process.

Output

z	Vector of log likelihoods.
_ht	Matrix of conditional variance.

Remarks The structural coefficients and the coefficients of the MGARCH process are estimated using maximum likelihood. The Multivariate GARCH model is given by:

$$h_t = C_0 + C_1\tilde{x}_t + \sum_{i=1}^{q} A_i\eta_{t-i} + \sum_{i=1}^{p} B_i h_{t-i}$$

where

$$\begin{aligned} y_{jt} &= f_j(x_t,\beta_j) + \epsilon_{jt} \qquad j = 1,\ldots,G \\ \epsilon_t &\sim N(0,H_t) \\ h_t &= \text{vech}(H_t) \\ \tilde{x}_t &= \text{vech}(x_t x_t') \\ \eta_t &= \text{vech}(\epsilon_t \epsilon_t') \end{aligned}$$

In general, there are G non-linear equations, with a residual vector ϵ_j for each equation. Based on these residuals, a conditional variance term, H_t is estimated. Two methods are available - the VEC formulation, and the BEKK formulation; H_t under BEKK should stay positive definite, while this is not necessarily the case under VEC.

The conditional variance h_t consists of four sets of terms; these are a constant (C_0), the parameters for the weakly exogenous variables (C_1), the ARCH

process (A_i), and the GARCH process (G_i). MGARCH-M can also be carried out, since the conditional variance, h_t, is available, and stored in each iteration under the global _HT. The order of each of these depends on the process:

Given G equations, there are $S = .5G(G + 1)$ elements to be estimated for each H_t. In addition, if there are J vectors x of weakly exogenous variables, there are $M = .5J(J + 1)$ product pairs, and thus M coefficients to be estimated.

Parameter Dimensions

	VEC		BEKK	
	Rows	Columns	Rows	Columns
ϵ	N	G	N	G
x	N	J	N	J
C_0	S	1	G	G
C_1	S	M	G	M
A_i	S	S	G	G
G_i	S	S	G	G

Note that for C_0 under BEKK, the GxG matrix is upper triangular. Each additional ARCH or GARCH term requires an additional A_i or G_i respectively - see the example below.

The residuals must be specified in a first set of FRMLs, and then the MGARCH process is specified in a second FRML. Note the use of the EQSUB command to simplify writing the likelihood. The estimation is rapid for a two equation system, where the GARCH term(s) are diagonal, since the process can be vectorized. For three or more equations, the conditional variance is derived recursively, which takes considerably longer.

The conditional variance (consisting of S time series) for the MGARCH processes is retrieved using the FORCST command, with MODE = CONDVAR. If no range is specified, the estimated conditional variance based on the actual residuals and estimated parameters is returned. If a range is specified, the estimated conditional variance is returned up to the first date of the range,

and the forecast based on the information up to the first date is returned for the period specified.

See the "General Notes for Non-Linear Models" under NLS, and the remarks under GARCH. An example is given in test19.prg.

Example

```
FRML ers1 e1 = y1 - b11 - b12*x1;
FRML ers2 e2 = y2 - b21 - b22*x2;
FRML elff llfn = mgarch(e1~e2,0,cvec,0,amt1~amt2,gmat);

FRML eqv1 cvec := c01|0|c03;
FRML eqv2 amt1 := a11~0~a13|0~0~0|a31~0~a33;
FRML eqv3 amt2 := diagrv(eye(3),b11|0|b33);
FRML eqv4 gmat := diagrv(eye(3),g11|0|g33);

PARAM c01 c03  a11 a13 a31 a33 b11 b33 g11 g33;
ML (p,i) ers1 ers2  elff;
   METHOD = bhhh bhhh nr;
   EQSUB = eqv1 eqv2 eqv3 eqv4;
FORCST cv11 cv12 cv22;
   METHOD = CONDVAR;
```

In this example, a system of equations is estimated with residual variance specified as simultaneous GARCH. Although not shown, it makes sense to model each equation separately to get initial starting values. The residuals are specified in ers1 and ers2, and the log likelihood is returned in elff. The model here is a GARCH(2,1) VEC process, without exogenous influences. Note how the parameters are specified as separate macros. The conditional variance is returned using the FORCST command.

Source

GARCHX.SRC

See Also

ARCH, EGARCH, FORCST, GARCH, ML, NLS

References

Engle, R.F., and K.F. Kroner (1995), "Multivariate Simultaneous Generalized ARCH", *Econometric Theory*, Vol. 11(1) pp. 122-150.

Purpose Estimates the coefficients of a user specified likelihood function.

Format
```
ML (options) elist ;
    BOUND = level ;
    EQCON = cnstrntlist ;
    EQSUB = macrolist ;
    GENALG = genalg ;
    GLOBOPT = globopt ;
    GRADIENT = &gradproc ;
    HESSIAN = &hessproc ;
    MAXIT  = maxit ;
    MAXSQZ = maxsqz ;
    METHOD = methname ;
    MODE  = modetype ;
    PENALTY = penalty ;
    POSDEF = pdname ;
    SIMANN = simann ;
    STEP = step ;
    TITLE = title ;
    TOL    = tolerance ;
    TRUST  = trust ;
    WEIGHT = wtname ;
```

Input

options	optional, print options.
elist	literal, required, equation list.
level	numeric, optional, percentage confidence level.
cnstrntlist	literal, optional, list of constraint equations.
macrolist	literal, optional, macro equation list.
genalg	numeric, optional, GA options (30,4 .4 .25).
globopt	numeric, optional, GO options (20000 100 .0001 4).
&gradproc	literal, optional, pointer to gradient procedure.
&hessproc	literal, optional, pointer to Hessian procedure.
maxit	numeric, optional, maximum number of iterations (20).
maxsqz	numeric, optional, maximum number of squeezes (10).
methname	literal, optional, algorithm list (BFGS BFGS BFGS).
modetype	literal, optional, mode list
penalty	literal, optional, penalty function (1000).

	pdname	literal, optional, positive definite algorithm (NG).
	simann	numeric, optional, SA options (5 .85 100 20).
	step	literal, optional, step type (LS).
	title	string, optional, title.
	tolerance	numeric, optional, param. convergence tolerance (.001).
	trust	numeric, optional, TR options (.1 1 .001 3).
	wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
LGCOEFF	Vector of Lagrangian coefficients.
GRADVEC	Gradient vector.
LLF	Log likelihood.
VCOV	Parameter covariance matrix.

Remarks

The ML command estimates the parameters of a model via the maximum likelihood method. The user specifies a FRML (or FRMLs) which computes the log-likelihood given a set of parameters. If a number of equations are specified, they are evaluated sequentially, and the output of the last equation is taken to be the log-likelihood. The form is very similar to NLS. Minimization of a function can be implemented by specifying MODE = MINIMIZE.

Four hill climbing estimation methods are available: BFGS, DFP, BHHH, and NR. For optimization problems with many local optima, one can use one of the direct search methods - GA, GO, NM and SA - as the second element of METHOD.

The user can optionally specify the name of a procedure for the gradient of the log-likelihood; this can be used by all the estimation methods. This reduces estimation time considerably. &*gradproc* is a pointer to a procedure written by the user; this procedure takes no arguments, and returns an *nxk* matrix where *k* is the number of coefficients to be estimated, in the same order as the order specified in the PARAM statement. The Hessian can likewise be specified by the user by specifying a pointer &*hessproc*. This is only used to

evaluate the Hessian if the method chosen is NR. This returns the sum of the *kxk* matrix of second differentials over *n* observations.

GAUSSX uses automatic differentiation if symgrad and/or symhess are used as the names for the gradient and/or Hessian procedures respectively. This requires that Maple 9 or higher is installed. See the Appendix for details.

Two step models can be estimated using the Murphy Topel variance correction by specifying the step for each estimation using the mode statement. For the first step, specify MODE = STEP1, and the second step MODE = STEP2.

See the "General Notes for Non-Linear Models" under NLS, and the examples given in test09.prg, test42.prg and test47.prg.

Example

1.
```
PARAM a1 a2 a3 ; value = .5 .5 .5;
FRML eq1 m = a1 + a2*x1 + a3*x2;
FRML eq2 llfn = y.*m - exp(m) - ln(y!);
ML (p,d) eq1 eq2;
   METHOD = bfgs bfgs nr;
ML (p,d) eq1 eq2;
   METHOD = bfgs bfgs nr;
   GRADIENT = &grd;
proc grd;
   retp(y-exp(m)).*(c~x1~x2) ;
endp;
```

2.
```
PARAM a1 a2 a3 sig; VALUE = 1 1 1 1;
FRML tob1 m = a1 + a2*z2 + a3*z3;
FRML tob2 llf1 = -(z1-m)^2./(2*sig) - .5*ln(2*pi*sig);
FRML tob3 llf2 = ln(cdfnc(m./sqrt(sig)));
FRML tob4 llfn = (z1 .gt 0).*llf1 + (z1 .le 0).*llf2;
ML (p,i) tob1 tob2 tob3 tob4;
   TITLE =  Tobit Model ;
```

3.
```
FRML ml1 xb1 = a0 + a1*exper + a2*educ + a3*white;
FRML ellf1 llfp = probit(employ,xb1,0);
```

```
FRML ml2 mr2 =ln( pdfn(-xb1)./cdfn(-xb1));
FRML ml3 resid = wage - (b0 +b1*educ + b2*exper
                 +  b3*fprof + b4*mr2);
FRML ellf2 llfn = normal(resid);
PARAM a0 a1 a2 a3;
ML (p,i) ml1 ellf1;
   METHOD = nr nr nr;
   MODE = step1;
   TITLE =  First step - Probit ;
CONST a0 a1 a2 a3;
PARAM b0 b1 b2 b3 b4 ;
ML (p,i) ml1 ml2 ml3 ellf2;
  METHOD = nr nr nr;
  MODE = step2;
  TITLE =  Second step - linear, MT correction ;
```

In example 1, a Poisson distribution is estimated. It makes it much easier to write the log-likelihood function (eq2) in terms of a variable, m, which is defined in eq1. The equations are estimated sequentially, so the order is important. The variable defined in the last equation (eq2) is taken as the log-likelihood (llfn). The display pauses (p) after each screen, and descriptive statistics (d) are displayed. Two estimations are shown. The first evaluates the gradient numerically, while in the second the user specifies a procedure for evaluating the gradient.

The second example shows how a Tobit model is estimated. Note how σ is incorporated as a parameter (sig). In this case, intermediate (i) results are displayed, and the output pauses (p) after each screen. A user specified title is shown.

The third example shows how a two step model is estimated In this example, a probit is estimated in the first step using equations ml1 and ellf1, This first step (of a two step estimation process) is characterized by specifying mode = step1. In the second step. a linear regression is estimated using a variate mr2 (a hazard function) derived conditional on the parameters specified in the first step. This second step, which uses equations ml1, ml2, ml3 and ml4 is characterized by specifying mode = step2. The Murphy Topel corrected

standard errors are displayed in the second estimation.

See Also FRML, NLS, TITLE, WEIGHT

References Amemiya, T. (1985), *Advanced Econometrics*, Harvard University Press, Cambridge, Mass.

Purpose Creates a vector of log likelihoods for a multinomial logit process.

Format *z* = MNL (*ycat, vmat*);

Input
ycat literal, vector of alternative chosen.
vmat literal, matrix of utility values for each alternative.

Output
z Vector of log likelihoods.

Remarks The structural coefficients and the coefficients of the MNL process are estimated using maximum likelihood. The multinomial logit model is based on the probability function

$$P_j = \frac{exp(U_j - U_k)}{\sum_j exp(U_j - U_k)}$$

where P_j is the probability of selecting alternative *j*, U_j is the utility associated with choice *j*, and U_k is the maximum utility over the possible choices. *ycat* is a vector in which is specified the alternative chosen for each observation. Each utility is specified in a FRML, and since utility differences are evaluated, the first utility is set to zero as a reference. *vmat* is the matrix formed by the concatenation of these utilities. The utilities can be functions of individual characteristics, (multinomial logit), choice characteristics (conditional logit), or a combination, and can be linear or non-linear.

See the "General Notes for Non-Linear Models" under NLS, and the discussion of linear MNL under QR. An example is given in test18.prg.

Example

```
FRML zp1 v1 = 0;
FRML zp2 v2 = g0 + g1*x1 + g2*x2;
FRML zp3 v3 = h0 + h1*x1 + g2*x3;
FRML zpmnl lllf = mnl(ycat,v1~v2~v3);
PARAM g0 h0 g1 g2 h1;
ML (p,i) zp1 zp2 zp3 zpmnl;
   TITLE =  Non-linear MNL ;
```

In this example, a linear mixed MNL model is estimated. `x1` is a individual characteristic, while `x2` and `x3` are choice based characteristics. `ycat` should

take values of 1, 2, or 3, depending on which alternative was selected for each observation.

Source GXPROCS.SRC

See Also ML, MNP, NLS, QR

References McFadden, D. (1976), " Conditional Logit Analysis of Qualitative Choice Behavior" in P. Zarembka, ed. *Frontiers in Econometrics*, Academic Press, New York.

Purpose

Creates a vector of log likelihoods for a multinomial probit process.

Format

z = MNP (*ycat, vmat, vcmat*);

Input

ycat literal, vector of alternative chosen.

vmat literal, matrix of utility values for each alternative.

vcmat

1. $.5K(K-1)$x1 vector of unique elements in the differenced covariance matrix.
2. KxK symmetric, positive definite covariance matrix of the K-variate normal density function.
3. KxK Cholesky factor of the KxK covariance matrix of the K-variate normal density function.
4. Kx(R+1) matrix for the factor analytic case, where covariance matrix has R factors.

_mnpsc global scalar, scaling option for *vcmat*, such that ||*vcmat*|| = 1. This helps precision somewhat. (Default = 1),

_mnpint global scalar, integration algorithm: 1 - analytical, 2 - simulation. (Default = 1).

MNP uses QDFN to evaluate the multivariate normal integral, and thus the following globals are used - see QDFN for documentation.

_qdfrep global scalar, the number of replications.

_qdfrlz global scalar, the number of realizations.

_qdford global scalar, the order of the integration

Output

z Vector of log likelihoods.

Remarks

The structural coefficients and the coefficients of the MNP process are estimated using maximum likelihood.

ycat is a vector in which is specified the alternative chosen for each observation. Each utility is specified in a FRML, and since utility differences are evaluated, the first utility is set to zero as a reference; *vmat* is the matrix formed by the concatenation of these utilities. The utilities can be functions of individual characteristics, (multinomial probit), choice characteristics (conditional probit), or a combination, and can be linear or non-linear.

The MNP procedure evaluates the probability of selecting the alternative specified in ycat. For each observation, the mean value (utility) associated with each alternative is stored in *vmat*. The Random Utility Model assumes that the utilities are distributed with the specified mean, and an additive disturbance that is correlated across alternatives. In the MNP formulation, the distribution of these errors is multivariately normal, with a covariance matrix Σ. For a K alternative model, there are $K^* = .5K(K-1)$ possible two choice combinations, and it can be shown that, after allowing for scaling, there can be no more than $K^* - 1$ free parameters in the covariance matrix. This covariance matrix, *vcmat*, can be entered in the following formats:

1. As a K^*x1 vector of parameters, with one held as a constant. No other restrictions are necessary.
2. As a KxK positive definite matrix, with $K^* - 1$ free parameters. Rank conditions must be satisfied.
3. As a KxK Cholesky decomposition of a PD matrix, stored as an upper triangular matrix, with $K^* - 1$ free parameters. Rank conditions must be satisfied.
4. As a Kx(R+1) matrix of factors for the factor analytic case. $\Sigma = D + BB'$. The first column is a vector of variances in the diagonal matrix D, and the remaining columns are the R factor loadings. See QDFN.SRC for a full description. Again there should be only $K^* - 1$ free parameters.

Integration of the multivariate density function is undertaken by QDFN. Exact estimation is the default, and is acceptably rapid for low K, or for the factor analytic case. For large K, simulation methods using the GHK algorithm are utilized. The QDFN globals must be set before a MNP estimation.

Since the utilities are estimated as differences, a reference is needed; usually this is achieved by setting the first utility equal to zero. Note also that identification is fragile in the MNP model without exclusion restrictions (Keane, 1992). The identification problem does not occur if the covariance matrix is specified, or if some explanatory variables do not occur in some of the utilities.

See the "General Notes for Non-Linear Models" under NLS, and the discussion of linear Probit under QR. An example is given in test18.prg.

Example

```
FRML zp1 v1 = 0;
FRML zp2 v2 = g0 + g1*x1 + g2*x2;
FRML zp3 v3 = h0 + h1*x1 + g2*x3;
FRML zpmnp lllf = mnp(ycat,v1~v2~v3,(sig1|sig2|sig3));
PARAM g0 h0 g1 g2 h1;
PARAM sig1 sig2 sig3;
   VALUE = 1 1 1;
   LOWERB = .0001 .0001 .0001;
CONST sig1;
ML (p,i) zp1 zp2 zp3 zpmnp;
   METHOD = nr bhhh nr;
   TITLE =  Non-linear MNP ;
```

In this example, a linear mixed MNP model is estimated. `x1` is a individual characteristic, while `x2` and `x3` are choice based characteristics. `ycat` should take values of 1, 2, or 3, depending on which alternative was selected for each observation. Since K is 3, K^*-1 is 2, and hence only two covariance parameters are free. In this example, the three parameters of the differenced covariance matrix (K^*) are specified, and one held constant for scaling.

Source

MNPX.SRC

See Also

FMNP, ML, MNL, NLS, QR

References

Greene, W.H. (1993), *Econometric Analysis*, 2nd ed. Macmillan, New York.

Hajivassiliou, V.A., D. McFadden, and P. Ruud. (1992), "Simulation of Multivariate Normal Orthant Probabilities: Methods and Programs", Cowles Foundation Discussion Paper No. 1021, Yale University, Conn.

Hausman, J.A., and D.A. Wise. (1978). "Conditional Probit Models for Qualitative Choice: Discrete Decisions recognizing Interdependence and Heterogeneous Preferences", *Econometrica,* Vol. 47, pp. 403-426.

Keane, M.P. (1992), "A Note on Identification in the Multinomial Probit Model". *Journal of Business & Economic Statistics*, Vol. 10 (2), pp.193-200.

Maddala, G.S. (1983), *Limited-dependent and Qualitative Variables in Econometrics*, Cambridge University Press, Cambridge.

Purpose Returns the modulus of the largest root of a vector or matrix.

Format *z* = MROOT (*phi*);

Input *phi* literal, vector or matrix.

Output *z* scalar, modulus

Remarks The stationarity conditions for a single equation AR process, with AR coefficients, ϕ, the roots of the characteristic equation:

$$C(z) = 1 - \phi_1 z - phi_2 z^2 - \cdots - phi_p z^p = 0$$

require a modulus greater than 1, or "lie outside the unit circle". For a VAR or VARMA process, a similar set of conditions hold.

For an AR(1) process, this is equivalent to the requirement that $|\phi_1| < 1$. MROOT provides an equivalent test, which can be used in both a single equation and multi equation contexts. This test returns a largest root (z) which is less than unity if the process is stationary.

Similarly, a process with MA coefficients θ is invertible if the largest root of θ has a modulus less than unity.

Example

```
FRML eq1 y =  arma(y, phi1|phi2, theta1|theta2);
FRML ec1 mroot(phi1|phi2) <= .9999;
FRML ec2 mroot(theta1|theta2) <= .9999;
NLS (p,i) eq1;
   OPLIST = constant;
   EQCON = ec1 ec2 ;
```

The ARMA process is estimated using constrained NLS, where the constraints impose both stationarity and invertibility on the AR and MA coefficients respectively.

Source TOOLSX.SRC

Purpose Creates a vector of log likelihoods for a Markov switching model.

Format *z* = MSM (*resid, sigma, prob, phi*);

Input

resid	literal, matrix of residuals.
sigma	literal, residual standard deviation parameters.
prob	literal, matrix of Markov transition parameters.
phi	literal, AR parameters.

Output

z	Vector of log likelihoods.
_mspm	Markov transition probabilities.
_msepv	Ergodic probability for full state vector.
_mseps	Ergodic probability for primitive states.
_msfp	Filtered probabilities.
_mssp	Smoothed probabilities

Remarks The MSM coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models. (MSM) allows a given variable to follow different time series processes over different subsamples. The choice of subsample is determined by a Markov process.

Assuming S states, the residual matrix *resid* will be an NxS matrix. The residuals can be derived from a linear or non-linear structural equation. The standard deviation for each residual is parameterized in *sigma* - thus *sigma* will be an Sx1 vector, or a scalar to restrict the same residual standard deviation across states.

The Markov transition matrix is specified as an (S-1)xS matrix of parameters - the last rows is determined residually so the probabilities sum to unity. To ensure positivity, the actual transition probabilities are derived from the square and norm of *prob*. The actual transition probabilities are available in *_mspm*.

Autoregressive terms are permitted - the order of the autoregressive structure is specified in *phi*. A value of zero implies no AR structure.

See the "General Notes for Non-Linear Models" under NLS, and the example under ML. An example is given in test36.prg.

Example

```
FRML eq1 res1 = y - m1;
FRML eq2 res2 = y - m2 - b1*delgnp;
FRML eq3 lllf = msm(res1~res2, sig1|sig2, p11~p12,
                    phi1|phi2);
ML (p,i) eq4 eq5 eq6;
   TITLE =  Markov Switching Model  ;
   METHOD = nr bfgs nr;
```

In this example, a two state model is estimated with an AR(2) structure. The first regime is simply a constant, while the second regime uses delgnp as a predictor of y.

Source

MSM.SRC

See Also

ML, NLS

References

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton University press, pp.685-689.

Purpose

Creates a vector of concentrated log likelihoods for a multivariate normal process.

z = MVN (*resid*);

Input

resid literal, matrix of residuals.

Output

z Vector of log likelihoods.

Remarks

The structural coefficients are estimated using maximum likelihood, under the assumption that the residuals are distributed multivariate normal. The concentrated likelihood is used. Note that it is assumed that the expected value of the residuals is zero.

This command permits the estimation of an equation or system of equations using ML instead of NLS. This can be useful in a two step process.

An example is given in test47.prg.

Example

```
FRML es1 rs1 = y1 - (a0 + a1*age + a2*educ + a3*mr2);
FRML es2 rs2 = y2 - (b0 + b1*sex + b2*mr2);
FRML ellf llf = mvn(rs1~rs2);
ML (p,i) eq1 es1 es2 ellf;
```

In this example, a system of equations is estimated using ML.

Source

GXPROCS.SRC

See Also

FRML, ML, NLS, NORMAL

Purpose

Creates a matrix of (pseudo) correlated random variables using specified distributions.

Format

s = MVRND (*n, k, dist, p, rmat, rtype*);

Input

n	scalar, number of observations.
k	scalar, number of variates.
dist	string or string array, distribution names.
p	*K*x4 or 1x4 matrix of parameters.
rmat	*K*x*K* correlation matrix, or scalar correlation coefficient.
rtype	scalar or character, correlation method.

Output

s	*N*x*K* matrix of correlated random variates.

Remarks

MVRND creates a matrix of correlated variates from specified distributions using copulas. *dist* is a string or string array consisting of distributions from the STATLIB library; if only a single distribution is specified, then each variate will be drawn from that distribution. *p* is a *K*x4 matrix of parameters, matching the distribution list; if *p* is a 1x4 vector of parameters, then this vector will be used for each variate.

Three correlation methods are available; the method is selected by specifying *rtype*:

[0 or 'p']	Pearson.
[1 or 'k']	Kendall Tau b.
[2 or 's']	Spearman Rank

MVRND is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
dist = "normal" $| "expon" $| "gamma";
let p[3,4] = 0 1 0 0   2 0 0 0   1.5 2.5 0 0;
let rmat[3,3] = 1 .5 .2 .5 1 .6 .2 .6 1;
s = mvrnd(1000, 3, dist, p, rmat, 2);
```

This example creates `s`, which is a 1000x3 matrix of correlated random variates consisting of the three distributions shown in `dist`, with the correlation structure specified by the Spearman rank matrix `rmat`.

Source COPULA.SRC

See Also COPULA, CORR, STATLIB

Purpose

Creates a vector of log likelihoods for a negative binomial process.

Format

z = NEGBIN (*y, indx1, indx2, trunc*);

Input

y	literal,	dependent variable - number of events.
indx1	literal,	index of the means - effect model.
indx2	literal,	index of the variance - dispersion model.
trunc	literal,	truncation vector.

Output

z	Vector of log likelihoods.	

Remarks

The coefficients of the two indices are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

The negative binomial model is a generalization of the Poisson model, The dependent variable, y is a non-negative integer specifying the number of events. As in the Poisson model, the distribution has a conditional mean λ_i, given by:

$$\lambda_i = \exp(\beta' x_i)$$

In addition, allowing for cross-section heterogeneity, the conditional variance (the dispersion model) is given by:

$$\sigma_i^2 = \lambda_i(1 + \exp(\gamma' z_i))$$

In the default, there is no truncation, and *trunc* is set to 0. If truncation occurs - for example y_i is always greater than zero for the number of crimes based on a prison inmate survey - then *trunc* is a two element vector consisting of the lower and upper truncation points.

See the “General Notes for Non-Linear Models” under NLS. An example is given in test56.prg.

Example

```
OLS wars age party unem;
PARAM a0 a1 a2;
   VALUE = coeff;
PARAM gamma0;
FRML eq1 indx = b0 +b1*age + b2*party + b3*unem;

1. FRML eq2 llf  = negbin(wars,indx,gamma0,0);
   ML (p,d,i) eq1 eq2;

2. FRML eq3 llf  = negbin(wars,indx,gamma0,1|15);
   ML (p,d,i) eq1 eq3;
```

In example 1, a linear negative binomial model is estimated, using OLS starting values. The RHS index is stipulated in eq1, and the log likelihood is returned from eq2. Example 2 does the same analysis, but with the assumption of a lower and upper truncation of 1 and 15 respectively.

Source GXPROCS.SRC

See Also ML, NLS, POISSON

References King, G.(1989), "Variance Specification in Event Count Models: From Restrictive Assumptions to a Generalized Estimator", *American Journal of Political Science*,, Vol. 33(3), pp. 762-784.

Purpose Provides workspace management when required.

Format NFACTOR = *size ;*

Input *size* numeric, required, scale size (1).

Remarks In the default mode, GAUSSX maintains a workspace using RAM, and NFACTOR is not used. For large data sets, or for student versions of GAUSS, workspace management will require disk storage, and this is specified in the GAUSSX CREATE statement.

When disk base workspace is specified GAUSSX automatically assesses the number of observations to read into core at any time based on the needs of the GAUSSX command currently being processed, and the available core. For the majority of cases, no user intervention is necessary. If GAUSS reports "*Insufficient Work Space*" for a particular command, it is possible for a user to alter the default memory allocation scheme, using the NFACTOR option. The default value for NFACTOR is unity; any value higher than this will inform GAUSSX to read fewer number of observations into core at each read.

NFACTOR operates only for the command specified. However, if an NFACTOR is given in the CREATE statement, it acts globally for all subsequent commands; any additional NFACTOR statements act only for the command in which they occur.

Example

```
NLS eq1 eq2 eq3 eq4 eq5;
   ENDOG = y1 y2 y3 y4 y5;
   NFACTOR = 1.5;
```

In this example, a system of equations is estimated by NLS, under default options. The value of NFACTOR should be increased heuristically.

See Also CREATE

Purpose Estimates the coefficients of a non-linear equation or system of equations.

Format
```
NLS (options) elist ;
    BOUND = level;
    EQCON = cnstrntlist;
    EQSUB = macrolist;
    GENALG = genalg;
    GLOBOPT = globopt;
    GROUP = grouplist;
    INST = instlist;
    MAXIT  = maxit;
    MAXITW  = maxitw;
    MAXSQZ = maxsqz;
    METHOD = methname;
    MODE  = modetype;
    NMA = nma;
    PENALTY = penalty;
    POSDEF = pdname;
    SIMANN = simann;
    STEP = step;
    TITLE = title;
    TOL  = tolerance;
    TRUST  = trust;
    WEIGHT = wtname;
    WINDOW = windowtype;
```

Input

options	optional, print options.
elist	literal, required, equation list.
level	numeric, optional, percentage confidence level.
cnstrntlist	literal, optional, list of constraint equations.
genalg	numeric, optional, GA options (`30 4 .4 .25`).
globopt	numeric, optional, GO options (`20000 100 .0001 4`).
macrolist	literal, optional, macro equation list.
grouplist	literal, optional, group variable list.
instlist	literal, optional, list of instruments.
maxit	numeric, optional, max. number of iterations (20 1).
maxitw	numeric, optional, max. covariance iterations (9999).

maxsqz	numeric, optional, max. number of squeezes (10).
methname	literal, optional, algorithm list (GAUSS GAUSS GAUSS).
modetype	literal, optional, mode list
nma	numeric, optional, list of moving average terms (0).
penalty	literal, optional, penalty function (1000).
pdname	literal, optional, positive definite algorithm (NG).
simann	numeric, optional, SA options (5 .85 100 20).
step	literal, optional, step type (LS).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001 0).
trust	numeric, optional, TR options (.1 1 .001 3).
wtname	literal, optional, weighting variable.
windowtype	literal/numeric, optional, spectral window.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
GRADVEC	Gradient vector.
LGCOEFF	Vector of Lagrangian coefficients.
LLF	Log likelihood.
VCOV	Parameter Covariance matrix.
COVU	Residual Covariance matrix.

Remarks

The NLS command estimates coefficients in single equations or systems of equations that are non-linear in their parameters. This is achieved iteratively by minimizing the sum of squares for the single equation, and by weighted least squares for systems of equations. For instrumental variables, a minimum distance estimator is used. For constrained optimization, this process is augmented using sequential quadratic programming.

General Notes

Specification The specified formulae in *elist* must be Type II—that is they must be expressed in terms of coefficients defined in a PARAM statement. Both the PARAM and the FRML commands must be specified before the GMM, NLS, ML, or FIML command.

Globals The variables specified as "Outputs" are returned as global variables. LGCOEFF is only returned for constrained optimization. The estimation can be repeated over groups using the GROUP option.

Print Options These include d—descriptive statistics, i—print intermediate iteration results, p—pause after each screen display, q—quiet - no screen or printed output, and s—symbolic -diagnostic output for AD.

Sample file Since non-linear estimation techniques usually require many passes through the data, a sample file is written containing just the observations corresponding to the current sample, and the variables required for the current estimation. Missing observations are listwise deleted from this subset. This option can be turned off using the argument NOSELECT on the OPTION command. In this latter case, no sample selection occurs (i.e. the sample is implied by the CREATE statement, and it is the user's responsibility to make sure that there are no missing values.

Iterations Iteration information is specified in *maxit*, which can consist of up to two elements. The first element specifies the maximum number of iterations. The second element, if specified, determines the frequency of the printing of the iterations.

Convergence Convergence is declared when the proportional change in each parameter is less than *tolerance* the default is 0.001. If *tolerance* consists of two elements, the first element represents the maximum proportional change in each parameter for convergence, and the second element represents the maximum proportional change in the objective function for convergence – convergence is achieved when either of these criteria is achieved. If convergence is not achieved within *maxit* iterations, estimation is terminated, and the current parameter results are displayed. Within each iteration, the stepsize is squeezed until there is an improvement in the objective function; the maximum number of such squeezes is given by *maxsqz*. These options can be changed at run time while the iterations are taking place by pressing any key, and then selecting the required option from the menu.

For non-linear estimations, convergence occurs in two iterations if a sin-

gle linear equation is being estimated. For non-linear systems, convergence is not guaranteed—if this happens, try different parameter starting values and/or different stepsize algorithms. The residual covariance matrix is updated at each iteration for system estimation under NLS or FIML. This can be controlled using the MAXITW option - as in non-linear SURE - see example 7.

Holding parameters The non-linear estimation methods - NLS, ML, GMM, and FIML - are used for estimating the parameters of a non-linear equation system. Given these estimated coefficients, one sometimes wishes to estimate the regression statistics (LLF, COVU etc) on a different data set, or a different sample, but with the same coefficients. This can be achieved by setting up the estimation in the normal way, but setting *maxit* = 0.

Gradient Finite differencing is the default method for evaluating the gradient and Hessian. Automatic differentiation can be specified for greater accuracy and speed. See the Appendix for details.

Macros Macros can be referenced within formulae used in non-linear estimation; this can be useful if one has a procedure that generates a matrix. See the discussion of macros under FRML and EQSUB. Note that such macros make the RECURS procedure described in previous versions of GAUSSX obsolete.

Algorithm Method The method to be used in determining the step size algorithm can be specified by the user. This is achieved by specifying a three element vector of names for *methname*, corresponding to the initial, remaining, and final iterations respectively. The available methods are:

- BFGS Broyden, Fletcher, Goldfarb and Shanno – an approximation of the Hessian is updated each iteration.
- BHHH The Berndt-Hall-Hall-Hausman algorithm; it gives exact maximum likelihood estimates of the standard errors, but convergence is slower than GAUSS.
- DFP Davidon, Fletcher and Powell – an approximation of the inverse Hessian is updated each iteration.
- DW The Dennis-Wolkowicz conjugate method may be superior to

BFGS or DFP.

GA Genetic algorithm mimics an evolutionary process by selecting those chromosomes that are fittest in the optimization context. See GENALG for details.

GAUSS The model is linearized in its variables and then estimated by multivariate regression applied to the reduced form. It gives exact maximum likelihood estimates, but incorrect standard errors.

GN A cross between the GAUSS and BHHH methods—the direction vector is weighted by the size of the residual.

GO Global optimization is a search algorithm which attempts to find a global optimum by a direct search over potentially optimal hypercubes. See GLOBOPT for details.

KILL Used in a script file to terminate an estimation. This option is useful if one wants to exclude a "*Failure to Improve*" iteration during a Monte Carlo simulation.

NM The Nelder-Meade method is a direct search algorithm using the downhill simplex method.

NR The Newton-Raphson algorithm estimates the Hessian matrix directly, while the other algorithms use an approximation. For large problems, this can take a long time.

ROBUST Heteroscedastic-consistent method - used for the final iteration only. This can take a very long time.

SA Simulated annealing is a search algorithm that attempts to find a global optimum by moving both up and downhill during the optimization process. See SIMANN for details.

If a "*Failure to Improve*" message is displayed, the user has a chance to change the method from the keyboard. For long jobs, it is sometimes desirable to set a script of what to do under this situation. The user can specify additional arguments to the METHOD option. These additional methods would them be implemented automatically – see example 6.

Step Type The step type method to be used. The available methods are:

[LS] Line Search—Search along the chosen direction using trial steps.

QP Quadratic Programming —Search using the quadratic programming algorithm. The QP algorithm is also used to determine step size when constraints are specified.

TR Trust Region—Search restricted to some region in the neighborhood of the current iterate.

Hessian The method to be used if the Hessian is not positive definite can be determined by the user by specifying *pdname*. The available methods are:

[NG] Newton-Greenstadt—Hessian forced positive definite by forcing all eigen values to be positive.

M Marquardt—ridge regression method.

QR Hessian evaluated using orthogonal triangular decomposition under BHHH, rather than inverting the gradient cross products. Data must fit in core.

Instruments Non-linear 2SLS and 3SLS is carried out by GAUSSX using the GMM algorithm. A list of instruments is specified using the INST option; these instruments are used for each equation in a system. Heteroscedastic consistent covariances (White) can be derived using the ROBUST option, and autocorrelated consistent covariances (Newey-West) using the WINDOW option. See GMM for a description of estimation methodology and robust standard errors.

Constraints Non-linear parameter constraints can be imposed under FIML, GMM, ML and NLS. Constrained non-linear programming is undertaken using sequential quadratic programming. The individual constraints are specified using the EQCON option, as well as from the upper and lower parameter bounds. The QP specification is obtained by linearizing the nonlinear constraints. The initial parameter values must be feasible. For each constraint, GAUSSX evaluates the value of the Lagrange multipliers, and these are stored as a GAUSS vector under the name LGCOEFF. The objective function gradient, which in the unconstrained case will be zero, is stored under GRADVEC. The confidence region for each parameter is derived at the confidence level specified in the BOUND option - typically *level* is set to 0.95. The reported bounds correspond to either the individual interval estimate based on the t-statistic, or the extreme value of the parameter which satisfies the joint (restricted) confidence

region at the specified level, whichever is the more restricted. See EQCON for an example.

For non-hill climbing methods - GA, GO, NM and SA, constraints are handled by adding a penalty function to the objective function, equal to the product of PENALTY and the violated constraint. PENALTY can be a scalar, or the name of a GAUSS vector with a length equal to the number of constraints. The default value is 1000.

Transfer functions These can be estimated using the NLS command. Autoregressive terms can be entered directly into the FRML using the LAG command - this allows both lagged endogenous as well as lagged exogenous variables to be specified. Each equation specified has a moving average process defined by the corresponding element in *nma* thus if three equations are to be estimated, then *nma* must have three elements. Differencing can be achieved either prior to estimation, using a GENR statement, or in the actual FRML. Estimation of a transfer function requires that all the vectors needed for the estimation must reside in core.

Two Step Estimation A two step process typically involves the use of a predicted value derived from parameters derived in a first step, which is then used as a variate in the second step; this results in biased estimates of the parameter standard errors. The Murphy Topel correction at the second step takes into account that the variate is a function of the first step parameters in order to obtain appropriate standard errors.

GAUSSX implements the Murphy Topel two step process for any two estimation processes using NLS and/or ML. The first step of such a two step process is characterized by setting *modetype* to step1, and the second step by setting *modetype* to step2. See ML for an example of a two step process.

Weighting This is available by directly specifying a weighted model, or by using the WEIGHT option – see the discussion under WEIGHT.

Examples of NLS are given in test02.prg. An example of a transfer function is given in test11.prg, an example of the use of simulated annealing is given in test21.prg, and an example of constrained estimation in test22.prg.

Example

```
PARAM a0 a1 a2 b0 b1 b2 c0 c1 c2;
FRML eq1 y1 = a0 + a1*x1 + x2^a2;
FRML eq2 y2 = b0 + b1*x3 + b2*x4;
FRML eq3 y2 = a0 + b1*lag(y2,1) + b2*x4;
FRML eq4 y4 = lag(y4,1) + c2*x2 + c2*x3;
```

1.
```
NLS  (p,d) eq1;
```

2.
```
NLS eq1 eq3;
```

3.
```
NLS  (p,i) eq1 eq2;
   MAXIT = 40;
   TOL  = .0001;
   METHOD = nr gauss robust;
FORCST y1fit y2fit;
```

4.
```
NLS  (p,i) eq3 eq4;
   NMA = 1 2;
```

5.
```
NLS  (p) eq3 eq4;
   INST = c x1 x3 z1 z2 z3;
```

6.
```
NLS eq1 eq2;
   METHOD = bhhh gauss bhhh nr bhhh dfp;
   TOL = 0  .000001;
```

7.
```
NLS eq1;  NLS eq2;
NLS eq1 eq2;
   MAXITW = 1;
```

8.
```
FRML eq1 q = a0*l^a1*k^a2;
FRML cq1 a1 >= 0;
FRML cq2 a2 >= 0;
FRML cq3 a1 + a2 <= 1;
PARAM a0 a1 a2;
NLS eq1;
   EQCON = cq1 cq2 cq3;
```

```
BOUND = .99;
```

In the first example, a single equation—eq1—is estimated using NLS. Execution pauses (p) after each screen display, and descriptive statistics (d) are displayed.

In the second example, cross-equation coefficient restrictions are imposed by specifying the same coefficient name in more than one equation.

Example 3 shows how a non-linear system of equations is estimated with a maximum of 40 iterations, and convergence to be declared when the relative proportional change for each parameter is less than .0001. The Newton-Raphson method is stipulated for the initial iteration, followed by GAUSS for the remainder, and the Heteroscedastic-consistent method for the final iteration. Execution pauses (p) after each screen, and the values of the parameters at each iteration (i) is displayed.

Example 4 shows a transfer function estimation - eq3 corresponds to an ARIMA(1,0,1) process, and eq4 to an ARIMA(0,1,2) process, but in each case there are additional regressors.

Example 5 demonstrates a non-linear 3SLS estimation process.

In example 6, a *failure to improve* situation would result in the GAUSS algorithm being replaced by one iteration using NR, one iteration using BHHH, and the remaining iterations using DFP. A subsequent *failure to improve* would replicate this process. Convergence is declared when the proportional change in the objective function is less than .000001;

Example 7 shows how a non-linear SURE estimation is carried out - the parameters are first estimated on each equations separately. Then a systems estimation is carried out, with a covariance matrix derived from the initial parameter estimates. In a linear system, this will generate the same parameter estimates as the SURE command.

Example 8 shows the estimation of a Cobb-Douglas production function with

additive error. The first two restrictions imply positive marginal products, while the third requires that the production function does not exhibit increasing returns to scale. The BOUND option generates a 99% confidence region for the restricted parameters.

See Also

EQCON, EQSUB, FIML, FRML, GMM, GROUP, ML, OPTION, TITLE, WEIGHT, WINDOW

References

Berndt, E.K., B.H. Hall, R.E. Hall and J.A. Hausman (1974), "Estimation and Inference in Nonlinear Structural Models", *Annals of Economic and Social Measurement*, Vol. 3/4, pp. 653-665.

Broyden, C.G. (1967), "Quasi Newton Methods and their Application to Function Minimization", *Math. Comp.*, Vol. 21, pp. 368-381.

Davidson, R., and J.G. MacKinnon (1993). *Estimation and Inference in Econometrics*, Oxford University Press, Oxford.

J.E.Dennis, J.E Jr. and H. Wolkowicz. (1993). Sizing and least-change secant methods *SIAM J. Numer. Anal.* Vol. 30, pp. 1291-1314.

Fletcher, R. (1980), *Practical Methods of Optimization*, Wiley, New York.

Flecher, R., and M.J.D. Powell (1963), "A Rapidly Convergent Descent Method for Minimization", *Computer Journal*, Vol. 6, pp. 163-168.

Goffe, W.L., G.D. Ferrier and J. Rogers (1994), "Global Optimization of Statistical Functions with Simulated Annealing", *Journal of Econometrics*, Vol. 60 (1/2), pp. 65-99.

Greenstadt, J. (1967), "On the Relative Efficiencies of Gradient Methods", *Mathematics of Computation*, pp. 360-367.

Jones, D.R, C.D. Perttunen, and B.E. Stuckman (1993), "Lipschitzian Optimization Without the Lipschitz Constant", *Journal of Optimization Theory and Application*, Vol 79(1), pp. 157-181.

Marquardt, D. (1963), "An Algorithm for Least-Squares Estimation of Nonlinear Parameters", *SIAM J. Appl. Math.*, Vol. 11, pp. 431-441.

Murphy, K. M. and R. H. Topel (1985). “Estimation and inference in two-Step Econometric Models”. *Journal of Business & Economic Statistics*, Vol 3(4), pp 370-379.

Nelder, J.A. and Mead, R. (1965). “A Simplex Method for Function Minimization”, *Computer Journal* Vol 7, pp. 308-313.

Powell, M.J.D. (1983), “Variable Metric Methods for Constrained Optimization”, *Mathematical Programming: The State of the Art* (A. Bachem, M.Grotschel and B.Korte, eds.), Springer-Verlag, pp. 288-311.

White, H. (1980), “A Hetroskedasticity-consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity”, *Econometrica*, Vol. 48, pp. 817-838.

Zellner, A., (1962), “An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests of Aggregation Bias”, *JASA*, Vol. 57, pp. 348-368.

Purpose Creates a vector with elements of unity if the argument element is not a missing value, else missing value.

Format *z* = NMV (*x*);

Input *x* literal, variable name.

Output *z* Vector, elements 1 or missing value.

Remarks The NMV command can be used within a GENR command. It creates a variable *z*, with elements equal to unity if the corresponding element in *x* is not a missing value, else the element in *z* is the GAUSS missing value code. This command would typically be used in data transformation when missing values occur, since the GAUSS relational operators do not return a missing value code when missing values are encountered. Note that GAUSSX operates with disable on, and missing values are listwise deleted from linear and non-linear estimations.

Example

```
GENR y2 = ( (y1 .le 3) + 2*(y1 .gt 3) ).*nmv(y1);
```

This example, shows how GAUSSX deals with missing values - `y2` is created by the two relational operations in the usual way; however to exclude missing values the result is multiplied by `nmv(y1)`. The net result for `y2` is the relational combination if `y1` is not a missing value, else a missing value.

See Also GENR

Purpose Preprocessor command to switch off the command file listing.

Format #NOLIST ;

Remarks Normally the entire GAUSSX command file listing is provided in the output file, prior to the execution listing. The command file listing can be selectively suppressed by using the #LIST and #NOLIST commands. #NOLIST; switches off the listing. #LIST; switches it back on.

Example

```
#NOLIST;
```

See Also #LIST, PAGE

Purpose

Transforms a vector so that it is normally distributed.

Format

NORMAL (*options*) *varname ;*
METHOD = *method;*
TITLE = *title;*
VLIST = *vlist;*

Input

options	optional, print options.
varname	literal, required, output variable name.
method	literal, optional, algorithm.
title	string, optional, title.
vlist	literal, required, input variable name.

Output

COEFF	coefficients.

Remarks

The NORMAL command transforms a vector so that it is normally distributed. Three algorithms are available; the algorithm is set in *method*:

BOXCOX	BoxCox transformation.
SNV	Standard normal variate (Default).
JOHNSON	Johnson transformation.

In each case, the data is standardized to zero mean and unit standard deviation. Under the SNV methodology, this is the only transformation undertaken.

The Boxcox transformation, $(x^{\lambda} - 1)/\lambda)$, selects λ optimally so as to maximize the normal probability plot correlation coefficient. This coefficient is stored in COEFF.

The Johnson transformation optimally selects one of the three families of distribution S: SL, SU, and SB, where L, U, and B refer to the variable being lognormal, unbounded, and bounded respectively. The selected distribution function is then used to transform the data to follow a normal distribution.

$$
\begin{aligned}
SL(1) \quad z &= \gamma + \eta * \ln(x - \varepsilon); \\
SU(2) \quad z &= \gamma + \eta * arcsinh((x - \varepsilon)/\lambda; \\
SB(3) \quad z &= \gamma + \eta * \ln((x - \varepsilon)/(\lambda + \varepsilon - x));
\end{aligned}
$$

The coefficients are stored in COEFF as a vector $\{S, \eta, \gamma, \lambda, \varepsilon\}$.

If the data is already normal, a warning is displayed, and the standardized variate (METHOD=SNV) is returned. The Anderson-Darling statistic is evaluated for the untransformed and transformed data. If a transformation cannot be carried out, a vector of missing values is returned.

Print options include b — print brief output only, p — pause after each screen display, and q — no screen display (quiet).

An example of NORMAL is given in test60.prg.

Example

```
NORMAL (p) ndta;
    METHOD = johnson;
    VLIST = dta;
```

This example transforms the vector dta using the Johnson transformation to a new vector ndta which follows a normal distribution.

Purpose Creates a vector of log likelihoods for a normal process.

Format *z* = *NORMAL* (*y, indx, pvec*);

Input

y	literal, dependent variable.
indx	literal, index of the means.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks The expected value of y_i is parameterized as:

$$E(y_i) = indx_i.$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β and *pvec*, are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The scale parameter - the standard deviation of *y* - must be positive. For linear models, the estimates are equivalent to `ols`.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale; value = 1;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn =  normal(fail,indx,scale);
   ML (p,i) eq0 eq1;

2  FRML eq2 llfn =  normal(fail~censor,indx,scale);
   ML (p,i) eq0 eq2;
```

In example 1, a normal model is estimated using maximum likelihood, with the index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, MVN, NLS

Purpose

Creates a vector of conditional means or density based on a nonparametric or semiparametric estimation.

Format

z = NPE (*y*, *x*, *h*);

Input

y	literal, response variable.
x	literal, kernel index.
h	literal or numeric, window width.
_npobs	global scalar, the observation number of the estimate. The estimate is calculated for all observations if _npobs is zero. (Default = 0).
_npcv	global scalar, validation type: 0 - no cross validation, 1 - cross validation. (Default = 0).
_npmth	global scalar, method: 0 - Fourier, 1 - direct. (Default = 0).
_npmod	global scalar, mode: 0 - conditional mean, 1 - density, 2 - discrete, 3 - smeared, 4 - frequency. (Default = 0).
_npper	global scalar, number of points in Fourier. (Default = 0).
_npkrn	global scalar, kernel: 0 - Gaussian, 1 - user defined. (Default = 0).
_nppnt	global scalar, print flag: 0 - do not print options, 1 - print options. (Default = 0).
_npout	global scalar, output flag: 0 - vector output, 1 - matrix output. (Default = 0).

Output

z	Vector, conditional mean or density.

Remarks

NPE (Non-Parametric Estimation) is a general nonparametric or semiparametric procedure that can be used within a GENR— statement, or as part of a GAUSS statement within a GAUSSX command file, or as part of a FRML statement. NPE estimates a univariate Gaussian kernel if *x* is a vector, or a multiplicative Gaussian kernel if *x* is a matrix. Thus the metric that is used to weight the observations in the kernel is the Euclidean distance. The Gaussian kernel for the *m*–variate index is given by:

$$K_h(x_i - x^*) = \frac{1}{(2\pi)^{.5m}} \prod_{j=1}^{m} e^{-.5\left(\frac{x_j^* - x_{ji}}{h}\right)^2}$$

where h is the window width, x_i is the index for observation i, and x^* is the reference index. Prior to being used in the kernel, the index is scaled to have unit variance, such that a single window width can be used. In the default, if x is a vector, the convolution of the data with the kernel will be undertaken using a fast Fourier transform (FFT); this is much faster than using direct calculation. All of the data must fit in core.

The window width, h, is crucial in kernel estimations, since it determines the amount of smoothing undertaken. If h is set to zero, h is set automatically at:

$$h = \left(\frac{4}{n(m+2)} \right)^{1/(4+m)}$$

where n is the number of observations, and m is the number of columns in x. This value for h is optimal for density estimation under certain conditions. In general, it is better to estimate h using cross-validation (see below).

In the default, the estimates are derived for all observations in the current sample. If y contains missing values, then these observations are not used in the kernel.

NPE within GENR

When used within a GENR statement, NPE will return the Nadaraya-Watson conditional mean for the current sample. The conditional mean, is given by:

$$\hat{m}(x_i) = \frac{\sum_{j=1}^{n} K_h(x_i - x_j) y_j}{\sum_{j=1}^{n} K_h(x_i - x_j)}$$

where the estimates are constructed from all the data points in the current sample, based on the window width h. If h is set to zero, a default value is used based on the current sample. The fast Fourier transform will be used if x is a vector; otherwise direct calculation (which is much slower) will be used. Program options can be set using the global variables prior to the GENR command.

Example

```
1.   GENR indx = a1*x1+a2*x2;
     GENR yhat1 = npe(y,indx,.4)

2.   GENR yhat2 = npe(y,x1~x2,0)
```

In the first example, indx is first defined for the current sample. yhat1 is the semiparametric estimate of the conditional mean of y, based on the Gaussian kernel with argument indx for window width 0.4. NPE will automatically scale indx to have unit variance. Since indx is a vector, the estimation will be undertaken using the FFT.

In the second example, the same analysis is undertaken, but this time using the multiplicative Gaussian kernel, with indices x1 and x2. The default window width is used, and since the index is a matrix, the conditional means are estimated using the direct method.

NPE within GAUSS

NPE can also be used in a GAUSS statement as part of a GAUSSX command file. This is useful if one wishes to change some of the defaults, or if a number of estimates are required, based on the same index. The following two examples give an indication of this type of use.

Example

```
SMPL 1 100;

1.  FETCH x1 x2 x3 y;
    y = y|0;
    x = x1~x2~x3;
    x = x|meanc(x)';
    _npobs = 101;
    yhat = npe(y,x,0);
    @@ yhat :  yhat;

2.  FETCH y indx;
    ydum = dummybr(y,1|2|3);
    _npcv = 1;
    _nppnt = 1;
    yhat = npe(ydum,indx,h);
    call makevars(yhat,0,yh1 yh2 yh3);
    STORE yh1 yh2 yh3;
```

In the first example, the conditional mean of y is estimated at the sample means of the index. y, x1, x2, and x3 are first fetched from the GAUSSX workspace under the current sample. y is augmented by zero, and the matrix x, consisting of x1 x2 x3, is augmented by the mean for each column. yhat is the conditional mean estimated only at observation 101 - i.e. at the sample mean. When _npobs is not zero, cross validation is used, and thus the zero element of y does not enter into the kernel, and hence yhat is estimated based only on the first 100 observations.

The second example shows how one would evaluate the nonparametric probability of falling in a particular categorical class, in a similar manner to the quantile response models. An index, indx and a categorical variable (y), which takes the values 1, 2, or 3, are fetched from the GAUSSX workspace. ydum, a 100x3 matrix, is created containing a value of unity in column k if $y_i = k$. Options can be specified by setting global variables prior to the estimation. yhat is the predicted probability based on the *leave-one-out* nonparametric estimation, using indx as the argument to the kernel. This is very efficient, since the convolution only has to be evaluated once for each observation, rather than three times. The three variables yh1, yh2, and yh3 are created from yhat, and stored in the GAUSSX workspace. Note that setting _npout = 1 would have created yhat as an nx1 vector of predicted probabilities for the observed category.

NPE within FRML

NPE can be used as part of a FRML statement to allow for an estimation of parameters, and/or window width. In the nonparametric case, the only parameter involved is the window width h, and this can be estimated using either least squares cross-validation (LSCV) in a regression context, or maximum likelihood cross-validation (MLCV) in a probability context. Similarly, the parameters of the index can be estimated with known (or default) h in a semiparametric context (without cross-validation), or both the index coefficients and the window width can be simultaneously estimated using LSCV or MLCV.

The maximum likelihood CV is given by:

$$\text{MLCV}(h) = n^{-1} \sum_{i=1}^{n} \log \hat{f}_{-i}(x_i)$$

where the *leave one out* density estimate $\hat{f}_{-i}(x_i)$ is constructed from all the data points except x_i :

$$\hat{f}_{-i}(x_i) = (n-1)^{-1} h^{-1} \sum_{j \neq i} K_h(x_i - x_j)$$

The window width, h is then chosen to maximize MLCV, using the ML command.

Least squares cross-validation uses the sum of squares as the appropriate criteria function. For density estimation, LSCV is defined as:

$$\text{LSCV}(h) = \sum_{i=1}^{n} (f(x_i) - \hat{f}_{-i}(x_i))^2$$

while in a regression context LSCV is:

$$\text{LSCV}(h) = \sum_{i=1}^{n} (y_i - \hat{m}_{-i}(x_i))^2$$

where the *leave one out* conditional mean $\hat{m}_{-i}(x_i)$ is defined as:

$$\hat{m}_{-i}(x_i) = \frac{\sum_{j \neq i} K_h(x_i - x_j) y_j}{\sum_{j \neq i} K_h(x_i - x_j)}$$

Again, the window width is chosen to minimize LSCV. In both cases, if a semi-parametric approach, such as projection pursuit, is being used, then the parameters of the index (x) can be estimated concurrently with h. LSCV is most conveniently estimated using the NLS command. The conditional expectation in this case can be retrieved using the FORCST command subsequent to the NLS estimation.

The program control options for both ML and for NLS are described in the "General Notes for Non-Linear Models" under NLS. In addition, there are some specific options available as subcommands when NPE is used within a FRML. The format of these options is:

OPLIST = *progopts ;*
ORDER = *weights ;*

```
PERIODS = periods ;
KERNEL = &kernproc ;
```

where:

progopts	literal, optional, options for NPE control.
weights	numeric, optional, smear weights. (.25, .5, .25)
periods	numeric, optional, #. of points in FFT.
&kernproc	literal, optional, pointer to kernel procedure.

Values in parentheses are the default values. The program control options are specified in *progopts*. The options available are:

`[NOCV]/CV` Specifies whether the estimation process is to use cross validation (*leave-one-out*) or not. Cross-validation is required if h is a parameter to be estimated.

`FOURIER/DIRECT` Specifies whether the convolution is to be estimated using the fast Fourier transform, or direct estimation. If the index has one column, the default is `FOURIER`. The number of points used in the FFT is given by *periods* the default is $2^m : 2^m > n$ where n is the sample size. If the index has more than one column, the estimation process will used `DIRECT`.

`[CM]/DENSITY/DISCRETE/SMEARED/FREQ` Specifies the type of estimation. CM returns an estimate of the Nadaraya-Watson conditional mean of the response variable y. `DENSITY` ignores y and returns the density $\hat{f}(x_i)$ for each point i; `FREQ` does the same, but without normalization, so that the totals do not sum to unity. `DISCRETE` takes a categorical variable $y : y \in \{1 \ldots m\}$, and returns the nonparametric probability of being in the observed category class for each observation. `SMEARED` does the same as `DISCRETE`, but assumes that the categories are ordered; the probability returned is the "smeared" probability over the neighboring categories. The default weighting is .25, .5, .25 centered on the observed category; the user can alter the weighting by specifying the elements in *weights* - there must be an odd number of elements.

`PRINT/[NOPRINT]` Specifies whether a description of the NPE options actually used should be printed out. This is useful for debugging.

Note that the program will over-ride specified options in certain cases.

NPE uses the Gaussian kernel by default. The user can specify an alternative kernel by specifying &*kernproc* this is a pointer to a procedure written by the user, and which takes the same arguments as proc gskernel. Only the DIRECT estimation method will be utilized in this context.

A number of examples of NPE estimation are given in test13.prg. The kernel estimator of the partial derivatives (the response function) are described under NPR (Nonparametric Regression).

Example

```
1.   PARAM a1 a2 a3; VALUE = .2 .4 .6;
     CONST a1;
     PARAM  h; VALUE = .5;
     FRML eq1 y = npe(y,a1*x1+a2*x2+a3*x3,h);
     FRML eq2 y = npe(y,x1~x2~x3,h);
     NLS (p,i) eq1;
        METHOD = bhhh bfgs bhhh;
        OPLIST = cv;
        PERIODS = 1024;
     FORCST yhat1;
     NLS (p,i) eq2;
        METHOD = bhhh bfgs bhhh;
        OPLIST = cv;
     FORCST yhat2;

2.   PARAM a1 a2 a3; VALUE = .2 .4 .6;
     CONST a1;
     FRML eq1 indx = a1*x1+a2*x2+a3*x3;
     FRML eq2 pr = npe(y,indx,0);
     FRML eq3 llfn = ln(pr.*(pr.>0) + .000001*(pr .<= 0));
     ML (i) eq1 eq2 eq3;
        OPLIST = discrete
```

Example 1 shows both a semiparametric and a nonparametric least squares

cross-validated estimation. In the first estimation, the FFT methodology is used, evaluated at 1024 points. Note the normalization required in the semiparametric case (a1 = .2), since the kernel evaluates the difference in the indices.

The second estimation uses the DIRECT methodology, since the index has three columns. The conditional means are evaluated in the FORCST statement, and stored in yhat1 and yhat2 respectively.

The second example shows how the parameters of a semiparametric estimation can be evaluated when the response variable y is categorical. A default window width is specified. The DISCRETE option ensures that pr is the semiparametric estimate of the probability for the observed category.

Source NPEX.SRC

See Also FORCST, FRML, GENR, ML, NLS, NPR

References

Härdle W. (1990), *Applied Nonparametric Regression*, Cambridge University Press, New York.

Klein, R.W., and R.H. Spady (1993), "An Efficient Semiparametric Estimator of the Binary Response Model". *Econometrica*, Vol. 61 (2), pp. 387-421.

Nadaraya, E.A. (1964), "On estimating regression", *Theory Prob. Appl.*, Vol.10, pp. 186-190.

Silverman, B.W. (1982), "Algorithm AS 176. Kernel density estimation using the fast Fourier transform", *Applied Statistics*, Vol. 31, pp. 93-97.

Silverman, B.W. (1990), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.

Watson, G.S. (1964), "Smooth regression analysis", *Sankhyā, Series A*, Vol. 26, pp. 359-372.

Purpose

Estimates the nonparametric regression statistics, and the response function at the sample mean.

Format

NPR (*options*) *vlist ;*
OPLIST = *progopts ;*
ORDER = *weights;*
PERIODS = *periods;*
REPLIC = *replicopts;*
TITLE = *title;*
WINDOW = *winwidth;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
progopts	literal, optional, options for NPR control.
weights	numeric, optional, smear weights. (.25, .5, .25)
periods	numeric, optional, #. of points in FFT.
replicopts	numeric, optional, replication options.
title	string, optional, title.
winwidth	literal or numeric, optional, window width.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
RSS	Residual sum of squares.
SER	Standard error of the regression.
LLF	Log likelihood.

Remarks

The NPR command carries out nonparametric or semiparametric regression. The normal use of NPR is to estimate the nonparametric conditional means, and the resulting residuals. This provides the information for printing the regression statistics. In addition, the response function, which are the partial derivatives of the conditional mean with respect to each of the kernel indices, are estimated at the sample mean along with their standard errors.

The kernel estimator of the conditional mean, y^*, conditional on x^* is:

$$\hat{E}(y^*|x^*) = \frac{\sum_{i=1}^{n} y_i K(x_i)}{\sum_{i=1}^{n} K(x_i)}$$

where $K(.)$ denotes the kernel based on the m variables in x. The product Gaussian kernel is used, since its properties are well established in the literature. See the discussion under NPE for the definition of the kernel, and the default window width. The kernel estimator of the jth partial derivatives (response function) at the point x^* is defined as:

$$\hat{\beta}_j(x^*) = \frac{\partial \hat{E}(y^*|x^*)}{\partial x_j} = \frac{\hat{E}(y^*|x_j^* + h/2) - \hat{E}(y^*|x_j^* - h/2)}{h}$$

When the kernel is Gaussian, $\hat{\beta}_j(x^*)$ can be evaluated analytically. These estimates are conditional on the choice of h, as well as any coefficient values used in a semiparametric kernel. These parameters should be optimally chosen using the NPE procedure, prior to running NPR.

The structure of the equation to be estimated is specified in the same manner as in OLS, using either a variable list, or a Type I FRML command. A constant is ignored. All observations must fit in core. Weighting is not applicable for NPR. The window width is specified in *winwidth* - the default size is described under NPE. The three commands OPLIST, ORDER and PERIODS are also described under NPE; these apply only to the estimation of the conditional means. The estimation of the response function takes place using a Gaussian kernel directly at the mean of the sample; this procedure is programmed for the conditional mean (CM), and is skipped if any other type of estimation is specified in the OPLIST command.

The value of the response function depends on the local conditions around the sample mean. A more robust measure can be derived using simulation techniques. The statement

REPLIC = *num nn*;

will estimate the response function from a random sample of size nn drawn with replacement from the current sample. This is repeated num times, and the means and standard errors of both the coefficients and the coefficient

standard errors are reported. These values will be available in COEFF and STDERR. If nn is not specified, the current sample size will be used.

Print options include p—pause after each screen display, d —print descriptive statistics, e —print elasticities, i —print estimates at each bootstrap iteration, b —brief summary statistics (faster) and q —quiet - no screen or printed output.

The type of forecast that is undertaken in a FORCST command will be determined, in the default, by the type of estimation undertaken in the preceding NPR command. These can be changed by using the nonparametric options - see the example. Unlike the parametric case, it is necessary to include all the past data when estimating future values of the response variable. Unknown values of the response variable should be set to missing, since elements with such values are excluded from the kernel.

See the "General Notes for Linear Models" under OLS, the "Remarks" under NPE, and the examples given in test13.prg.

Example

```
1.   NPR y x1 x2 ;

2.   PARAM a1 a2 a3; VALUE = 1 .4 .6;
     CONST a1;
     PARAM  h; VALUE = .5;
     FRML eq1 y = npe(y,a1*x1+a2*x2+a3*x3,h);
     NLS (p,i) eq1;
        OPLIST = cv;
        PERIODS = 1024;
     GENR indx = a1*x1 + a2*x2+ a3*x3;
     NPR (d,p) y indx;
        WINDOW = h;
        PERIODS = 1024;
        OPLIST = print;
        REPLIC = 20 60;


3.   SMPL 1951 1980;
```

```
FRML eq1 y x1 x2 x2(-1);
NPR (i) eq1;
   REPLIC = 100;
SMPL 1951 1990;
FORCST yhat;
   OPLIST = print;
```

In example 1, a nonparametric regression is carried out with y as the response variable, and c, x1 and x2 as the explanatory variables. The default window width will be calculated. The conditional means will be estimated using the DIRECT method, with no cross-validation. Two response coefficients will be calculated.

In the second example a semiparametric regression is undertaken. The coefficients in the index, as well as the window width are estimated using the NPE command, using cross-validation. The index indx is then generated based on these coefficients. The NPR command uses this window width in the WINDOW statement. Note both the NPE and the NPR use the FOURIER method, with 1024 points. The regression statistics will differ however, since the NPR estimate does not specify cross-validation. The partial derivative estimates at the sample mean are based on a simulation of 20 draws, with each draw using 60 observations.

The third example shows how a Type I formula can be used in an NPR command. In this case, a nonparametric regression is carried out using a default window width, and the simulation estimate of the partial derivatives is based on 100 draws using 40 (1951-1990) observations per draw as the default. The values at each draw will be shown since the (i) option is specified. The conditional mean is created by the FORCST command, with a print option. If there are only missing values for y for 1981 to 1990, then the conditional means for the entire sample will be based on the response variables for 1951 to 1980.

See Also FORCST, OLS, NPE, TITLE, WINDOW

References Rilstone, P., and A. Ullah (1989), “Nonparametric Estimation of Response

Coefficients", *Communications in Statistics*, Vol. 18, pp. 2615-2627.

Ullah, A. and H.D. Vinod (1988), "Nonparametric Kernel Estimation of Econometric Parameters", *Journal of Quantitative Economics*, Vol. 4 (1), pp. 81-87.

Purpose Returns the observation number for a particular date.

Format *z* = NUMDATE (*x*);

Input *x* literal, date.

Output *z* Vector, observation number.

Remarks The NUMDATE command can be used within a GENR command. It creates a variable *z*, with elements numbered relative to the first date in the workspace. The command can be used to create a trend variable when *x* is the vector _ID. Note that gaps in the current sample result in discontinuities in *z*.

Example

```
CREATE (q) 19701 19794;
SMPL 19711 19794;
GENR trend = numdate(_ID);
```

This example, shows how a trend variable can be produced. In this case, trend will take values for 19711 to 19794 of 5 through 40; the values for 19701 to 19704 will be missing.

See Also DUMMY, GENR

Purpose

Estimates the coefficients in an equation using ordinary least squares.

Format

OLS (*options*) *vlist ;*
METHOD = *methname;*
GROUP = *grouplist;*
PDL = *pdllist;*
TITLE = *title;*
WEIGHT = *wtname;*
WINDOW = *windowtype;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
methname	literal, optional, covariance method (NONE)
grouplist	literal, optional, group variable list.
pdllist	literal, optional, options for PDL.
title	string, optional, title.
wtname	literal, optional, weighting variable.
windowtype	literal/numeric, optional, spectral window.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
DF	Degrees of freedom.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
LLF	Log likelihood.
RSQ	R-squared.
RBARSQ	RBAR-squared.
VCOV	Parameter covariance matrix.

Remarks

The OLS command carries out classical ordinary least squares.

General Notes

Linear Models

The structure of the equation to be estimated can be specified either by using a list of variables, with the dependent variable first, as in example 1 below, or by using an equation name which has been previously specified in a Type I FRML command. This is true for all single equation models (AR, ARCH, KALMAN, OLS, PANEL, PLS, NPR, POISSON, QR, ROBUST, and 2SLS).

The variables described in "Outputs" are returned as global variables; they can subsequently be used in any GAUSS or GAUSSX command. Grouped output is available using the GROUP option. Print options include p—pause after each screen display, d—print descriptive statistics, e—print elasticities, i—print parameters at each iteration, q—quiet - no screen or printed output, s—print diagnostic statistics and v—print parameter covariance matrix.

Diagnostics are included for single equations with the s option. These include Godfrey's test of residual serial correlation, Durbin-Watson test of positive and negative residual serial correlation, Ramsey's RESET test of functional form, Jarque-Bera test of residual normality, Lagrange multiplier test for heteroscedasticity, and a Chi-squared test for parameter stability. For instrumental variables, Sargan's test of misspecification is produced. Diagnostics are not available when GROUP is specified. Additional information is available through the on-line help (Alt-H).

Lagged variables can be used by specifying the lag in parenthesis - see example 2. Polynomial distributed lags can be specified using the PDL option.

Elasticities ($\beta_i \bar{X}_i / \bar{Y}_i$), evaluated at the sample mean, or weighted sample mean if weighted regression, are available if the e print option is specified.

Weighted regressions are available using the WEIGHT option. Heteroscedastic-consistent variance-covariance matrix of parameters, corrected for the degrees of freedom, is available by setting *methname* to ROBUST; the default is NONE; the summary statistics are based on the method used. The Newey-West procedure provides estimators whose variance has also been corrected for autocorrelated disturbances, in addition to heteroscedasticity; both the spectral window (weight structure) and the maximum lag length are defined in the WINDOW command.

Example

```
1.   OLS y c x1 x2 ;

2.   OLS (d,p,s) y c x1 x2(-1);
        WEIGHT = wtname;

3.   FRML eq1 y c x1 x2;
     OLS eq1;
        METHOD = robust;
```

In example 1, an OLS is carried out with y as the dependent variable, and c, x1 and x2 as the independent variables. Note that c is a vector of unity which allows a non-zero intercept.

A similar regression is estimated in example 2, in which x2 is replaced with its lagged value, but in this case the variables are weighted using the elements of wtname, descriptive statistics (d) and diagnostic statistics are produced (s), and execution pauses (p) after each screen display.

In example 3, OLS is performed on the structural equation specified in eq1, and a heteroscedastic consistent covariance matrix is computed.

See Also

FRML, PDL, TITLE, WEIGHT, WINDOW

References

Chow, G.C. (1960), "Tests of equality between sets of coefficients in two linear regressions", *Econometrica*, Vol. 28, pp. 591-605.

Godfrey, L.G. (1978), "Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables", *Econometrica*, Vol. 46, pp. 1293-1302.

Godfrey, L.G. (1978), "Testing for higher order serial correlation in regression equations when the regressors include e lagged dependent variables", *Econometrica*, Vol. 46, pp. 1303-1310.

Greene, W.H. (1993), *Econometric Analysis*, 2nd ed., Macmillan, New York.

Jarque, C.M., and A.K. Bera (1980), "Efficient tests for normality, homoscedas-

ticity and serial independence of regression residuals", *Economic Letters*, Vol. 6, pp. 255-259.

Koenker, R. (1981), "A note on studentizing a test for heteroskedasticity", *Journal of Econometrics*, Vol. 17, pp. 107-112.

Newey, W.K., and K.D. West (1987), "A Simple Positive Semi-Definite Heteroskedasticity and Autocorrelation Consistent Covariance Matrix" *Econometrica*, Vol. 55, pp. 703-708.

Ramsey, J.B. (1969), "Tests for specification errors in classical linear least squares regression analysis", *Journal of the Royal Statistical Society* B, Vol. 31, pp. 350-371.

Ramsey J.B. (1970), "Models, specification error and inference: a discussion of some problems in econometric methodology", *Bulletin of the Oxford Institute of Economics and Statistics*, Vol. 32, pp. 301-318.

White, H. (1980), "A Hetroskedasticity-consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity", *Econometrica*, Vol. 48, pp. 817-838.

Purpose

To read an external data file into GAUSSX .

Format

OPEN (*options*) *vlist ;*
FNAME = *filename ;*
FMTLIST = *fmtopts ;*
RANGE = *rangelist ;*
OPLIST = *progopts ;*

Input

options	optional, print options.
vlist	literal, optional, variable list.
filename	literal, required, the name of an external file.
fmtopts	literal, optional, format options.
rangelist	literal, optional, spreadsheet range.
progopts	literal, optional, options for program control.

Remarks

The OPEN statement reads the specified file from disk using the path specified in the DATA path option in the GAUSSX desktop. Data is read into GAUSSX starting at the first value specified in the CREATE statement, and is independent of the current SMPL. The number of observations to be read in must be less than the range specified in the CREATE statement, and must be of the same type.

Spreadsheet and ASCII (delimited and packed) files are supported as described below. GAUSSX for Windows supports a larger range of formats, and hence it is preferable to use the data exchange facility from the GAUSSX desktop to create GAUSS data files from foreign files prior to running the command file.

The following file formats are supported; the file extension tells GAUSSX which format to use.

DAT	GAUSS or GAUSSX data file
FMT	GAUSS matrix file
WKS	Lotus 1-2-3, revision 1-A
WK1	Lotus 1-2-3, revision 2.x
WK3	Lotus 1-2-3, revision 3.x
WK4	Lotus 1-2-3, revision 4.x

WRK Lotus Symphony, version 1.0
XLS Excel, versions 2.0 to Excel 2003
* ASCII file

GAUSS/GAUSSX files If the file to be opened is a GAUSS or GAUSSX data file, *vlist* need consist only of the stem; the .DAT subscript is not required. The default is to read all the variables on the file; a subset is read if the variable list is specified. The names of the variables in the file will be printed out if the d print option is specified, and a pause after each screen if the p option is specified.

GAUSS matrix files The subscript .FMT must be used, and the variable list is required.

ASCII files A filename with an extension that is not specifically given above is treated as an ASCII file. The variable list is required, unless the first row contains delineated headings, in which case the headings are used to name the variables (not applicable on UNIX platforms).

```
@ country  gnp    pop @
    1     1200    640
    2     2100    820
```

For space delineated ASCII files, each column becomes a GAUSSX vector. Packed ASCII files can also be read; these files have fixed record length, but there are no delineators between elements. The option FMTLIST specifies the record length, and field position, length and floating point position for each element – see FMTLIST for additional information and examples. A GAUSS data file of the same name and with a .DAT extension will be created, and that file will be used for future reads as the default - see *progopts* below. Note that if you have an ASCII file with a .DAT extension, the file will be overwritten by a GAUSS file with the same name.

Spreadsheet files These are recognized by the appropriate extension. Note that neither Lotus SQZ files nor Excel Workbooks are supported – save the files as standard worksheets. The variable list is required. Each column in the spreadsheet becomes a GAUSSX vector. In the

default, the entire spreadsheet is read in; ranges however are permitted - the range *rangelist* can either be a named range, or a set of cell coordinates. Only numeric input is permitted. Data is read into GAUSSX starting at the first value specified in the CREATE statement, and consequently there must not be more observations than specified in the CREATE range. The variable list is required, unless the first row in the spreadsheet contains headings, in which case the headings are used to name the variables. Otherwise, the number of names declared in the OPEN statement must equal the number of columns read in. A GAUSS data file of the same name and with a .DAT extension will be created, and that file will be used for future reads as the default - see *progopts* below.

The program control options are specified in *progopts*. The options available are:

REPLACE/[NOREPL] Specifies whether an existing GAUSS data file is to be replaced by a new file created by ATOG. Thus an ASCII to GAUSS conversion will not take place if a GAUSS data file exists with the same prefix as the file specified in *fname*, unless the REPLACE option is specified.

An example is given in test04.prg.

Example

```
1.   OPEN; FNAME = gsxfile;

2.   OPEN (d,p) x1 x2 x3; FNAME = gsxfile;

3.   OPEN z1 z2 z3; FNAME = country.asc;

4.   OPEN x1 x2; FNAME = gsfile.fmt;

5.   OPEN x1 x2 x3;
        FNAME = data.asc;
        FMTLIST = record=24 position=1 width=8;

6.   OPEN (p) x1 x2;
```

```
FNAME = spread.wks;
RANGE = A1 B60;
```

In example 1, a GAUSS or GAUSSX data set is opened, and all the vectors present are loaded into GAUSSX . Note that the .DAT and .DHT extensions are not used. Both gsxfile.dta and gsxfile.dat must be in the directory specified in the DATA path of the GAUSSX desktop.

In example 2, only vectors x1, x2 and x3 are read into GAUSSX . A listing of the names of the variables in gsxfile is produced under the d option.

In example 3, country.asc is an ASCII file, which GAUSSX converts into the gauss data file country.dat. In this example, both the extension (.asc) and the vector list (z1, z2, z3) are required. If the data file contains '@' delineated headers, then the vector list is not required.

Example 4 shows the same rule for GAUSS matrix files - only the first two columns are read into x1 and x2 respectively.

In example 5, three fields are read from an ASCII file with record length of 24 (excluding final carriage return and line feed). Each field is 8 characters wide.

Example 6 shows how a range can be specified to input a certain block of data. The p option produces a pause to allow a report on the number of cells read.

See Also CREATE, DROP, FMTLIST, KEEP, SAVE

Purpose Sets GAUSSX options

Format OPTION *oplist ;*

Input *oplist* literal, required, option list.

Remarks The OPTION command allows a number of GAUSSX options to be changed during a program. Some of these options can be initially set in the GAUSSX menu—the OPTION command overrides these initial settings. The options are:

AD/[FD]	Gradients and Hessians are evaluated symbolically / numerically.
DEBUG/[NODEBUG]	Debug facility is enabled / disabled.
EXCEL/[NOEXCEL]	Excel process for import/export of .xls files.
GPLOT/[PQG]	Graphic support using GAUSSPlot / Publication Quality Graphics.
GRADH =	Set default perturbation value for numerical evaluated gradients (default = 0.000001).
[GRAPH]/TEXT	Graphic routines use graphic / text display.
INCORE/[NOINCORE]	Data must / need-not fit in core.
MAXLAG =	Set maximum lag length (default = 12).
MAXLINES =	Set maximum number of lines printed per page. (default = 40).
MONO/[COLOUR]	Graphic output in monochrome / colour mode.
[OUTPUT]/NOOUPUT	Output to output device is turned on / off.
[OUTW80]/OUTW132	Set output width to 80 / 132 columns.
PRINT/[NOPRINT]	Output file is LPT1 / defined in desktop.
[REPL]/NOREPL	Default data transformation sets vector to missing value for range outside of current sample. Under NOREPL, data transformations do not affect vector outside of current sample.
[SCREEN]/NOSCREEN	Turns screen on / off.
[SELECT]/NOSELECT	A sample file is written / not-written before estimation.
SINGLE/[DOUBLE]	Files written in single / double precision.
[WARN]/NOWARN	GAUSSX warnings are enabled / disabled.

Example

```
OPTION outw132 print noscreen maxlag=24;
```

In this example, all output is sent to the printer, at 132 column format, the screen is turned off, and the maximum lag is set at 24 periods.

Purpose

Creates a vector of log likelihoods for an ordered logit model.

Format

z = ORDLGT (*ycat*, *xmat*);

Input

ycat	literal, vector of alternative chosen.
xmat	literal, matrix of utility values for each alternative.

Output

z	Vector of log likelihoods.

Remarks

The structural and threshold coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

An example is given in test08.prg.

Example

```
PARAM t0 t1 t2 t3;
   VALUE = -30 1 2 30;
CONST t0 t3;
FRML eq1 xb = (t0~t1~t2~t3) - (a1*x1 + a2*x2);
FRML eq2 llf = ordlgt(y,xb);
ML (p,i) eq1 eq2 ;
   METHOD = bhhh bhhh nr;
```

This example estimates a three choice ordered logit model. Since there are three alternatives, two threshold parameters are required (t1 and t2, as well as the bounds at minus and plus infinity respectively.

Source

GSPROCS.SRC

See Also

ML, QR

Purpose

Creates a vector of log likelihoods for an ordered probit model.

Format

z = ORDPRBT (*ycat, xmat*);

Input

ycat	literal,	vector of alternative chosen.
xmat	literal,	matrix of utility values for each alternative.

Output

z	Vector of	log likelihoods.

Remarks

The structural and threshold coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

An example is given in test08.prg.

Example

```
PARAM t0 t1 t2 t3;
   VALUE = -30 1 2 30;
CONST t0 t3;
FRML eq1 xb = (t0~t1~t2~t3) - (a1*x1 + a2*x2);
FRML eq2 llf = ordprbt(y,xb);
ML (p,i) eq1 eq2 ;
   METHOD = bhhh bhhh nr;
```

This example estimates a three choice ordered probit model. Since there are three alternatives, two threshold parameters are required (t1 and t2, as well as the bounds at minus and plus infinity respectively.

Source

GSPROCS.SRC

See Also

ML, QR

Purpose

To force a page break on the output file.

Format

PAGE ;

Remarks

This command places a form-feed symbol on the output file. It can be used before a command to force the output from that command to be placed on a new page when printed. This works for word processors such as Word and Wordpad, but not for editors, such as Notepad.

This, and other page formatting controls can also be achieved by using the GAUSS special characters (see lprint in the GAUSS manual). Note that the @@ syntax is required to turn output to ON.

Example

```
1.   PAGE ;

2.   @@ \f \e \69;
```

In example 1, a page break is produced. In example 2, a page break is produced (\f), followed by an escape sequence (Escape E). Thus this capability provides a facility for font control, spacing, etc.

See Also

GAUSS, OPTION

Purpose

Estimates the coefficients of a linear regression model for panel data.

Format

PANEL (*options*) *vlist ;*
IDENT = *identifier;*
METHOD = *methname;*
MODE = *modetype;*
TITLE = *title;*
WEIGHT = *wtname;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
identifier	literal, required, panel identifier name.
methname	literal, optional, variance components. (SWAMY)
modetype	literal, optional, modeltype. (FIXED)
title	string, optional, title.
wtname	literal, optional, weighting variable.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
DF	Degrees of freedom.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
VCOV	Parameter covariance matrix.

Remarks

The PANEL command estimates the coefficients of a linear regression model for panel data. The data can be balanced (all groups have the same number of individuals) or unbalanced.

The structure of the equation to be estimated can be specified either by using a list of variables, with the dependent variable first, as in example 1 below, or by using an equation name which has been previously specified in a Type I FRML command.

Print options include b — brief output, d — print descriptive statistics, p — pause after each screen display, and q — quiet - no screen or printed output,

PANEL uses the series specified in *identifier* to identify each individual. Thus, for example, if there 90 observations consisting of 6 firms with 15 observations for each firm, then there needs to be a firm identifier series that takes the same value for each firm.

```
CUSIP  DATE     x1     x2

2791   1991    2.3    4.2
2791   1992    2.5    4.4
2791   1993    3.0    4.7
          ...
2791   2004    4.7    5.7
2791   2005    4.8    5.5
3441   1991    1.7    2.2
3441   1992    1.9    2.4
          ...
```

In this example, the firms are identified using:

```
IDENT = cusip;
```

There are two basic frameworks used by the PANEL regression model; these are specified in *modetype*:

[FE]	Fixed Effects — This model adds group specific constant terms to the regression model. This model is also known as the LSDV (least squares dummy variable) model.
RE	Random Effects — The individual specific constant is assumed randomly distributed across cross-sectional units. This model is also known as the Error Components model.

There are a number of methods for estimation of the variance components in the random effects model; these are specified in *methname*:

AMEMIYA	Amemiyia (1971).
NERLOVE	Nerlove (1971). This method is always used for unbalanced data.
[SWAMY]	Swamy and Auora (1972). This is the default method.
WALLACE	Wallace and Hussein (1969).

The variables described in "Outputs" are returned as global variables; they can subsequently be used in any GAUSS or GAUSSX command.

See the "General Notes for Linear Models" under OLS, and the example given in test58.prg.

Example

```
1.   PANEL (p) y c x1 x2 ;
        IDENT = cusip;

2.   FRML eq1 cost c capital labour;
     PANEL (p,d) eq1;
        IDENT = firmid;
        METHOD = nerlove;
        MODE = re;
```

In example 1, a PANEL is carried out with y as the dependent variable, and c, x1 and x2 as the independent variables. This is (by default) a fixed effects regression; the number of intercepts equals the number of groups specified in the cusip vector. The constant c is automatically dropped. Note that c is a vector of unity which allows a non-zero intercept.

In example 2, a PANEL estimation is performed on the cost function specified in eq1. This is a random effects model using the Nerlove variance components method, with the groups specified in firmid.

See Also ANOVA, FRML, OLS, TABULATE, TITLE, WEIGHT

References

Baltagi, B.H. (2001), *Econometric Analysis of Panel Data*. John Wiley and sons. ltd.

Purpose

Provides a starting value for parameters specified in non-linear formulae used by GMM, FIML, ML and NLS.

Format

PARAM *plist ;*
LOWERB = *lvalues;*
ORDER = *order;*
RANGE = *range;*
SYMBOL = *rootname;*
UPPERB = *uvalues;*
VALUE = *values;*

Input

plist	literal, required, parameter list.
lvalues	numeric, optional, lower bounds.
order	numeric, optional, matrix order.
range	numeric, optional, submatrix range.
rootname	literal, optional, element name.
uvalues	numeric, optional, upper bounds.
values	numeric, optional, starting values.

Remarks

The PARAM statement adds the variables in *plist* to the list of GAUSSX parameters, updates the value of the parameters if *values* is specified, and creates global symbols for each parameter in *plist*, initialized at the current value. After an estimation, the parameters will no longer be at their starting value, but will retain the coefficient estimates of the last non-linear estimation.

Parameters must be initialized before estimating an equation in which such parameters appear. If *values* is not specified, each parameter in *plist* is given a default value of zero. If *values* is the name of a global variable, then the elements of this vector will be used as the starting value. The number of elements in *plist* and *values* must be the same. *values* can also be the name of a vector. Thus following a linear estimation, the coefficient values are stored in a vector called COEFF. These values can be used to set the values for a set of parameters by setting *value* equal to COEFF. Note however that the number of elements in COEFF must be the same as the number of terms in *plist*.

The values specified by *lvalues* and *uvalues* are lower and upper limits which constrain the parameter in subsequent non-linear estimation procedures. If

an option is specified, the number of arguments must match the number of parameters specified. These parameter constraints are imposed as a wall during the estimation process, mainly to ensure that a parameter does not move into a non-feasible region. A more general approach is to use constrained optimization - see EQCON.

A matrix of parameters can be created by specifying a single matrix name in *plist*, and the *rootname* of the elements in SYMBOL. In this case, either the row and column order must be specified in ORDER, or *values* must be the name of a predefined matrix of the required values and order. Once a matrix of parameters has been specified, sub-blocks can be altered using the VALUE and RANGE options. RANGE is a four element vector specifying the desired sub block – the initial row, initial column, final row, and final column. The order of the matrix specified in VALUE must match the block specified in RANGE. The sub-block specified is changed, while the remaining elements are not altered.

Example

```
1.   PARAM a0 ;

2.   PARAM b0 b1 b2;
        VALUE = .3 0 -.2;

3.   PARAM co c1;
        LOWERB = 0 0;
        UPPERB = 10 10;

4.   OLS y c x1 x2 x3;
     PARAM a0 a1 a2 a3;
        VALUE = coeff;

5.   PARAM amat;
        SYMBOL = a;
        ORDER = 4 3;
     FRML eq1 y = mproc(x1~x2~x3~x4,amat);
     NLS eq1;
     avec = ones(1,3);
     CONST amat;
```

```
   VALUE = avec;
   RANGE = 2 1 2 3;
NLS eq1;
```

In example 1, a single parameter is specified. If a0 had previously been defined as a constant, it maintains its previous value; if not, its value is set to zero.

In the second example, starting values are specified by use of the VALUE option.

The third example shows how lower and upper bounds can be imposed on parameters.

In example 4, the coefficients from the previous regression are stored as a vector (COEFF); in this case a0 will be given the value of the intercept, a1 the coefficient on x1, etc.

Example 5 shows how a 4x3 matrix of coefficients is created (amat), with elements a_{ij}. In the subsequent estimation, 12 coefficients will be estimated. After the estimation, the second row of AMAT is set to unity as a constant, and eq1 is then re-estimated.

See Also ANALYZ, CONST, FRML

Purpose Creates a vector of log likelihoods for a Pareto process.

Format *z* = PARETO (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, shape index.
pvec	literal, location parameter.

Output

z	Vector of log likelihoods.

Remarks The Pareto distribution was historically used to describe the allocation of wealth among individuals. The Pareto distribution has two parameters - location and shape. However, the maximum likelihood estimate of the location parameter is simply the minimum of *y*. Thus only shape is estimated. Typically, the expected value of shape ($E(s_i)$) is parameterized as:

$$E(s_i) = \exp(indx_i)$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
FETCH wealth;
   v1 = minc(wealth);
   v2 = 1/meanc(ln(wealth/v1));
   x0 = v2|0;

PARAM b0 b1 ;
   value = x0;
CONST loc; value = v1;
FRML eq0 shape = b0 +b1*income

1  FRML eq1 llfn =  pareto(wealth,shape,loc);
   ML (p,i) eq0 eq1;
      eqcon = ec1;

2  FRML eq2 llfn =  pareto(wealth~censor,shape,loc);
   ML (p,i) eq0 eq2;
```

In example 1, a Pareto model is estimated using maximum likelihood, with the shape index defined in `eq0`, and the log likelihood in `eq1`. The location parameter is not estimated using `pareto`, since it is the minimum of `y`. Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Computes the probability density function for the specified distribution.

Format

y = PDF (*pdfname, x, p1, p2, p3*);

Input

pdfname	string, the name of the probability distribution.
x	*N*x*K* matrix, the argument to the specified distribution.
p1	*N*x*K* matrix or scalar, first parameter for the specified distribution.
p2	*N*x*K* matrix or scalar, second parameter for the specified distribution.
p3	*N*x*K* matrix or scalar, third parameter for the specified distribution.

Output

y	*N*x*K* matrix of probabilities.

Remarks

This procedure returns the probability density for the specified distribution.

General Notes

Probability Density Functions

The probability density functions, means and variances of the supported distributions are shown in the following tables. Each distribution is characterized by parameters - *p1*, *p2* and *p3*. These parameters must be conformable with *x* - either the same size, or scalar. If a distribution uses only one parameter, *p2* and *p3* are set to zero. If a distribution uses two parameters, then *p3* is set to zero. Note that the χ^2, *F*, and Student's *t* distributions can be specified with a non-centrality parameter, which is defined in the same manner as the respective GAUSS CDF functions.

The probability density function is specified in *pdfname*. The following are supported:

BETA — The beta pdf. takes an argument, *x*, which must lie in the interval [0 1], and two parameters, *a* and *b*, both of which must be positive.

BINOM — The binomial pdf takes an integer, non-negative argument, *x*, and two parameters, *n*, which is a positive integer, and *p*, which must lie in the interval [0 1]. *y* is the probability of *x* successes in *n* independent trials, where *p* is the probability

Distribution	PDF
Beta	$f(x\|a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$
Binomial	$f(x\|n,p) = \left(\begin{array}{c} n \\ x \end{array} \right) p^x (1-p)^{n-x}$
χ^2	$f(x\|v) = \frac{1}{\Gamma(.5)}(.5)^{.5v} x^{.5(v-2)} e^{-.5x}$
Cauchy	$[\pi b(1 + (fracx - ab)^2]^{-1}$
Exponential	$f(x\|\lambda) = \frac{e^{-x/\lambda}}{\lambda}$
F	$f(x\|v_1,v_2) = \frac{\Gamma(.5(v_1+v_2))}{\Gamma(.5v_1)\Gamma(.5v_2)} \left(\frac{v_1}{v_2}\right)^{(v_1/v_2)} \frac{x^{.5(v_1-2)}}{\left[1+\left(\frac{v_1}{v_2}\right)x\right]^{.5(v_1+v_2)}}$
Gamma	$f(x\|a,b) = \frac{1}{b^a\Gamma(a)} x^{a-1} e^{-x/b}$
Geometric	$f(x\|p) = p(1-p)^x$
Gumbel	$(1/b)e^{(a-x)/b} e e^{(a-x)/b)}$
Hypergeometric	$f(x\|m,k,n) = \left(\begin{array}{c} n \\ x \end{array} \right) \left(\begin{array}{c} m-k \\ n-x \end{array} \right) \div \left(\begin{array}{c} m \\ n \end{array} \right)$
Logistic	$\mathrm{sech}^2[(x-a)/2b]/4b$
Log-Normal	$f(x\|\mu,\sigma^2) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-.5(\ln(x)-\mu)^2/\sigma^2}$
Neg. Binomial	$f(x\|s,p) = \left(\begin{array}{c} x+s-1 \\ s-1 \end{array} \right) p^s (1-p)^x$
Normal	$f(x\|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-.5(x-\mu)^2/\sigma^2}$
Laplace	$.5\frac{e^{-\|x-a\|/b}}{b}$

Distribution	PDF
Pareto	ba^b/x^{b+1}
Poisson	$f(x\|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$
T	$f(x\|v) = \frac{\Gamma(.5(v+1))}{\Gamma(.5v)}\frac{1}{\sqrt{\pi v}}\frac{1}{(1+x^2/v)^{.5(v+1)}}$
Uniform	$f(x\|a,b) = \frac{1}{b-a}$
Weibull	$f(x\|a,b) = abx^{b-1}e^{-ax^b}$

of success in any given trial.

CAUCHY — The Cauchy pdf takes an unbounded argument, *x*, and two parameters, p1, the median, and p2, a positive scale parameter. It has no moments. It is infinitely divisible, since the mean of *n*-independent Cauchy distributions is also Cauchy.

CHISQ — The Chi-squared pdf takes a non-negative argument, *x*, and a single parameter, *v*, the degree of freedom, which must be a positive integer. A second optional positive scalar noncentrality parameter can be specified. The sum of squares of *v* observations, each independently distributed standard normal, is distributed chi-squared with *v* degrees of freedom.

EXP — The exponential pdf takes a non-negative argument, *x*, and a single parameter, λ, the mean, which must be positive. The exponential function is used to model waiting times.

F — The F pdf takes a non-negative argument, *x*, and two parameters, *v*1 and *v*2, both of which must be positive integers. A

Distribution	Mean	Variance
Beta	$\frac{a}{a+b}$	$\frac{ab}{(a+b+1)(a+b)^2}$
Binomial	np	$np(1-p)$
Cauchy	None	None
χ^2	v	$2v$
Exponential	λ	λ^2
F	$\frac{v_2}{v_2-2}$ [for $v_2 > 2$]	$\frac{2v_2^2(v_1+v_2-2)}{v_1(v_2-2)^2(v_2-4)}$ [for $v_2 > 4$]
Gamma	ab	ab^2
Geometric	$(1-p)/p$	$(1-p)/p^2$
Gumbel	$a + .5772b$	$b^2\pi^2/6$
Hypergeometric	nk/m	$nk(m-k)(m-n)/m^2(m-1)$
Laplace	a	$2b^2$
Logistic	a	$(\pi b)^2/3$
Log-Normal	$e^(\mu + .5\sigma^2)$	$e^{2\mu+\sigma^2}(e^{\sigma^2} - 1)$
Neg. Binomial	$s(1-p)/p$	$s(1-p)/p^2$
Normal	μ	σ^2
Pareto	$ab/(b-1)$	$a^2b/((b-1)^2(b-2))$
Poisson	λ	λ

Distribution	Mean	Variance
T	0 [for $v > 1$]	$\frac{v}{v-2}$ [for $v > 2$]
Uniform	$(a+b)/2$	$(b-a)^2/12$
Weibull	$a^{-1/b}\Gamma(1+b^{-1})$	$a^{-2/b}\left[\Gamma(1+2b^{-1}) - \Gamma^2(1+b^{-1})\right]$
Wishart	ab	$2ab_{jj}^2$

third optional positive scalar non-centrality parameter can be specified.

`GAMMA` The gamma pdf takes a non-negative argument, x, and two parameters, a and b, both of which must be positive. The gamma distribution is typically used in reliability models.

`GEOM` The geometric pdf takes a non-negative integer argument, x, and a single parameter, p, which must lie in the interval [0 1]. y is the probability of x failures before a success, where p is the probability of success in any given trial.

`GUMBEL` The Gumbel (or extreme value) pdf takes an argument, x, and two parameters, p1, the mode, and p2, a positive scale parameter. The Gumbel distribution is used in the derivation of the MNL model.

`HYGEOM` The hypergeometric pdf takes a non-negative integer argument, x, and three positive integer parameters, m, k, and n. If there exist k objects of a certain type out of a total of m objects, and n objects are drawn at random without replacement, then y is the probability of drawing exactly x items of the specified type.

LAPLACE The Laplace pdf takes an unbounded argument, x, and two parameters, μ, the mean, and b, a positive scale parameter.

LOGISTIC The logistic pdf takes a positive argument, x, and two parameters, p1, the mean, and p2, a positive scale parameter.

LOGNORM The log-normal pdf takes a positive argument, x, and two parameters, μ and σ^2, the mean and variance of the associated normal pdf. The variance must be positive. If y is log-normal, then $\ln(y)$ is normal. It is used for variates which can only take positive values, such as the size of particles in an emulsion.

NEGBIN The negative binomial pdf takes an integer, non-negative argument, x, and two parameters, s, which is a non-negative integer, and p, which must lie in the interval [0 1]. y is the probability of x failures before the sth success, where p is the probability of success in any given trial.

NORMAL The normal pdf takes an unbounded argument, x, and two parameters, μ, the mean, and σ^2, the variance, which must be positive. Note that the normal density function supports both univariate and multivariate distributions. A multivariate distribution is recognized if *p2* is square and has the order K, where K is the column size of x. In the multivariate case, the mean *p1* can be scalar, Kx1, or NxK, and *p2* must be positive definite.

NORMALTL The left truncated normal pdf takes three parameters, μ, the mean, σ^2, the variance, which must be positive. and v, the left truncation point. This distribution is used in Bayesian analysis for data augmentation, for example in a tobit model.

NORMALTR The right truncated normal pdf takes three parameters, μ, the mean, σ^2, the variance, which must be positive. and v, the right truncation point. This distribution is used in Bayesian analysis for data augmentation, for example in a tobit model.

PARETO The Pareto pdf takes an argument, x ($x > p1$), and two parameters, p1, a positive location parameter, and p2, a positive scale parameter.
“ttfamily “bfseries “upshape LaTeX Error: There’s no line here to end.ΩΩSee the LaTeX manual or LaTeX Companion for explanation.ΩType ‘H ’return ¿ ‘for ‘immediate ‘help

PEARSON The Pearson pdf takes an argument, x, and three parameters, c_0, c_1, and c_2. This distribution is very general, and includes as special cases the beta, gamma, normal and t distributions. This family is modelled assuming zero mean. The standard normal distribution corresponds to $c_0 = 1$, $c_1 = 1$, and $c_2 = 0$.
“ttfamily “bfseries “upshape LaTeX Error: There’s no line here to end.ΩΩSee the LaTeX manual or LaTeX Companion for explanation.ΩType ‘H ’return ¿ ‘for ‘immediate ‘help

POISSON The Poisson pdf takes a non-negative integer argument, x, and a single positive parameter, λ, the mean. y is the probability of x events occurring within a period, where λ is the expected number of events in that period.

T The Student’s t pdf takes an unbounded argument, x, and a parameter, v, the degrees of freedom, which is a positive integer. A second optional parameter (*p2*) can be specified for the covariance matrix for a multivariate t-distribution; this matrix must be square and of order K, where K is the column size of x. A third optional positive scalar non-centrality parameter (*p3*) can be specified for the univariate case. The Student’s t distribution tends to the normal distribution as $v \rightarrow \infty$.

UNIFORM The uniform pdf takes an argument, x, which must lie in the interval [a b], and two parameters, a and b, where b must be greater than a. y has the same probability at each point in the specified interval.

WEIBULL The Weibull pdf takes a non-negative argument, x, and two positive parameters, a and b. The type 1 extreme value distribution is derived from the Weibull distribution.

WISHART The Wishart pdf takes two parameters, n, the degrees of freedom, and Σ, a positive definite covariance matrix. This distribution is used in Bayesian analysis to model the posterior distribution for Σ conditional on structural parameters θ.

PDF is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
x = seqa(0,.2,6);
a = 2; b = 4;
p = pdf(beta,x,a,b,0);

 x' =  0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

 p' =  0.0000  2.0480  1.7280  0.7680  0.1280  0.0000
```

This computes the probability given the argument x and parameters a and b for the beta pdf.

Source PDFX.SRC

See Also CDF, CDFI, QDFN, RND, STATLIB

References

Abramowitz, M., and I. Stegun (1972), *Handbook of Mathematical Functions*, Dover Publications, New York.

Devroye, L. (1986), *Non-Uniform Random Variate Generation*, Springer-Verlag, New York.

Evans, M., N. Hastings and B. Peacock (1993), *Statistical Distributions*, 2nd ed. John Wiley, New York.

Johnson, N.L. and S. Kotz (1970). *Distributions in Statistics: Continuous Univariate Distributions - 1*. John Wiley & Sons, New York.

Press, W.H. *et. al.* (1986), *Numerical Recipes*, Cambridge University Press, New York.

Purpose

Generates a PDL (polynomial distributed lag) variable of right hand side variables for Type I equation.

Format

GAUSSX COMMAND *vlist;*
PDL = *pdllist;*

Input

vlist literal, required, variable list.
pdllist literal, required, options for PDL.

Remarks

The PDL options are available for any non-probabilistic Type I estimation and are specified in *pdllist*. For each PDL variable specification, *pdllist* has the form:

vname nper plag zcon

vname is the name of the RHS variable which is to be lagged according to the PDL specification.

nper is the number of terms in the polynomial; thus this is the degree of the polynomial plus one.

plag is the number of lags of the variable to be included, which includes the zero lag; thus this is the maximum lag plus one.

zcon is the end point constraint. This forces the coefficients at either end of the lags to be set to zero. `NEAR` forces the first lead to zero, `FAR` forces the *plag* lag to be zero. `BOTH` imposes both of these constraints, and `NONE` imposes no end point constraints.

The number of coefficients estimated is equal to *nper* less the number of constraints. This must be less than or equal to *plag*. This format must be repeated for each PDL variable; the estimated coefficients are called `A000001, A000002...` for the first variable, `B000001, B000002...` for the second, etc. The "unscrambled" coefficients are presented for each PDL variable.

If the sample period starts at the beginning of the workspace, the first *plag-1* observations are dropped. The PDL option can be used on systems of

equations, and on instrumental estimation - in each case any appearance of the variable specified in *vname* is replaced with its "scrambled" form. This applies to RHS variables and to instruments.

Example

```
1.   OLS (p,d) y c x1 x2 x3 x4;
        PDL = x1 3 3 FAR
              x2 2 4 NONE;

2.   FRML eq1 y1 c x1 x2 ;
     FRML eq2 y2 c x1 x3 x4 ;
     3SLS (p,d) eq1 eq2;
        INST = c x1 x3 z1 z2 z3;
        PDL  = x1 2 4 NONE;
```

In the first example, a PDL estimation is carried out for variables x1 and x2 using OLS. x1 is specified as being a polynomial of degree two covering lags up to and including $x1_{t-2}$, with a coefficient of zero on the third lag, while x2 is specified as a polynomial of degree one with no restrictions covering lags up to $x2_{t-3}$.

The second example shows a 3SLS estimation in which the variable x1 is specified as a polynomial of degree two with no end-point restrictions covering lags up to $x2_{t-3}$. This will apply to x1 in both equations, as well as in the list of instruments. Note that in the case of instrumental variables, care must be taken that the number of instruments is at least as large as the maximum number of coefficients to be estimated in each equation.

See Also

AR, ARCH, LAG, OLS, SURE, VAR, 2SLS, 3SLS

References

Almon, S. (1965), "The Distributed Lag between Capital Appropriations and Expenditures", *Econometrica*, Vol. 33, pp. 178-196.

Purpose Returns the smallest root for a set of correlation coefficients.

Format *z* = PDROOT (*rho*);

Input *rho* literal, required, correlation coefficient vector

Output *z* scalar, smallest root.

Remarks PDROOT creates the correlation matrix from the correlation coefficients *rho*; this matrix is positive definite if all its characteristic roots are positive. PDROOT returns the smallest root. Thus if this root is greater than zero, the correlation matrix is positive definite.

Example

```
FRML eq1 xb1 = a0 + a1*x1 + a2*x2;
FRML eq2 xb2 = b0 + b1*x3 + b2*x4;
FRML eq3 xb3 = c0 + c1*x1 + c2*x4;
FRML ellf3 llf = probit(y1~y2~y3,xb1~xb2~xb3,r12|r13|r23);
FRML ec1 r12^2 <= .9999;
FRML ec2 r13^2 <= .9999;
FRML ec3 r23^2 <= .9999;
FRML ecpd pdroot(r12|r13|r23) >= .0001;
ML (p,i) eq1 eq2 eq3 ellf3;
   EQCON = ec1 ec2 ec3 ecpd;
   TITLE = Trivariate Probit;
```

This example estimates a trivariate probit model, restricting the correlation coefficients to lie in the prescribed range.

Source TOOLSX.SRC

Purpose

Creates a vector of log likelihoods for a Pearson process.

Format

z = PEARSON (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, location index.
pvec	literal, two element parameter vector (scale |shape).

Output

z	Vector of log likelihoods.

Remarks

The Pearson distribution has been used to model data that exhibited skewness - which typically includes survival data, which are often asymmetric. The expected value of location is parameterized as:

$$E(y_i) = (indx_i)$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β, of the index are estimated using maximum likelihood; thus this can be used for linear or non-linear models.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1;
  VALUE = sv;
PARAM scale shape;
  VALUE = 1 1;

FRML eq0 indx = b0 + b1*x1 + b2*x2;

1 FRML eq1 llfn =  pearson(wealth,indx,scale|shape);
  ML (p,i) eq0 eq1;
     eqcon = ec1;

2 FRML eq2 llfn =  pearson(wealth~censor,indx,scale|shape);
  ML (p,i) eq0 eq2;
```

In example 1, a Pearson model is estimated using maximum likelihood, with the location index defined in `eq0`, and the log likelihood in `eq1`. Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

Creates a vector of log likelihoods for a power GARCH process.

Format

z = PGARCH (*resid, avec, bvec, gvec*);
z = PGARCH_T (*resid, avec, bvec, gvec, dvec*);

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for the ARCH process.
bvec	literal, vector of parameters for the GARCH process.
gvec	literal, γ and δ parameters.
dvec	literal, distributional parameter (ν).

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The structural coefficients and the coefficients of the PGARCH process are estimated using maximum likelihood. The PGARCH model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ h_t &= \alpha_0 + \sum_{i=1} \alpha_i (|\epsilon_{t-i}| - \gamma \epsilon_{t-i})^{\delta} + \sum_{j=1} \beta_j h_{t-j} \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The α are the vectors of the weights for the lagged asymmetric ϵ^2 terms; this is the ARCH process. The β are the weights for the lagged h terms; this is the GARCH process.

avec is a vector of parameters giving the weights for the lagged asymmetric squared residuals. The first element, which is required, gives the constant. *gvec* is a two element vector of parameters for the asymmetric process consisting of γ and δ. *bvec* is the vector of parameters for the GARCH process.

Note the stationarity conditions described under GARCH.

See the "General Notes for GARCH" under GARCH, and the "General Notes for Non-Linear Models" under NLS.

Example

```
OLS y c x1 x2;
sigsq = serˆ2;
PARAM c0 c1 c2;
   VALUE = coeff;
PARAM a0 a1 a2 b1 g1 d1;
   VALUE =  sigsq .1 .1 0 0 2;
FRML cs1 a0 >= .000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 b1 >= 0;
FRML cs5 a1+a2+b1 <= .999999;
FRML eq1 resid = y - (c0 + c1*x1 + c2*x2);
FRML eq2 lf = pgarch(resid,a0|a1|a2,b1,g1|d1);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4 cs5;
```

In this example, a linear PGARCH model is estimated using constrained maximum likelihood, with OLS starting values. The residuals are specified in eq1, and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive. Setting d1 as a constant to get initial starting values facilitates estimation.

Source GARCHX.SRC

See Also GARCH, EQCON, FRML, ML, NLS

References Ding, Z., R.F. Engle, and C.W.J. Granger. (1993), "A Long Memory Property of Stock Market Returns and a New Model", *Journal of Empirical Finance*, Vol 1 (1), pp 83-106.

Purpose

Plots one or more series against time (_ID).

Format

PLOT (*options*) *vlist ;*
FNAME = *filename;*
GROUP = *grouplist;*
MODE = *mode;*
SYMBOL = *symlist;*
TITLE = *title;*
VLIST = *vname;*

Input

options	optional, print options.
vlist	literal, required, variable list.
filename	literal, optional, macrofile.
grouplist	literal, optional, group variable list.
mode	literal, optional, graph mode (LINE).
symlist	literal, optional, symbol list.
title	string, optional, user defined title.
vname	literal, optional, x-axis variable (ID).

Remarks

The PLOT command will produce a plot of one or more series against _ID. The _ID series is time if the CREATE command uses annual data, and is the number of the observation if the CREATE command uses quarterly, monthly or undated data. An alternative series can be specified in *vname*. Graphing by groups is available using the GROUP option.

Print options include p – pause until the graphic is closed, m – display for five seconds (PQG), h – print graph, and r – rotate graph (PQG).

See "General notes for Graphs" in GRAPH. Examples of PLOT are given in tutor.prg and in test53.prg.

Example

```
1.   PLOT  x1 x2;

2.    OPTION PQG;
      _pbox = 1;
      PLOT (p) x1 x2 x3;
        VLIST = x4;
        TITLE = Graph 1;

3.   OPTION GPLOT;
     PLOT (p) x1 x2;
        FNAME = test3.mcr;

4.   OPTION GPLOT;
     PLOT (p) x1 x2;
        SYMBOL = 1 5  1 4   2 1;
```

In the first example, observations for x1 and x2 are plotted against time under the current sample.

In the second example, a graphic screen is displayed using PQG in which x1, x2, and x3 are plotted against x4, The execution pauses (p) until the graph is closed. A box is drawn round the screen, and the user defined title is displayed.

Examples 3 and 4 show how a graphic display is customized using GAUSS-Plot. The first, and more powerful method is to use a macro file, as shown in example 3. (See example 4 under GRAPH for details).

Example 4 shows how some details can be set using the SYMBOL command. Three characteristics can be set for each variable plotted as a line or symbol - color, shape/pattern and size. Since there are 2 variables (x1 and x2), there will be 6 elements. In this case the lines are drawn in black and green (1 5), as solid and dotted (1 4) and with thickness of 2 and 1.

See Also GRAPH, GROUP, OPTION, TITLE

Purpose

Estimates the coefficients in an equation using partial least squares.

Format

PLS (*options*) *vlist ;*
ORDER = *maxfactor;*
MAXIT = *maxpress;*
TITLE = *title;*
TOL = *tolerance;*
WEIGHT = *wtname;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
maxfactor	numeric, required, maximum number of factors.
maxpress	numeric, optional, maximum number of factors for cv (20).
title	string, optional, title.
tolerance	numeric, optional, factor variation tolerance.
wtname	literal, optional, weighting variable.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
DF	Degrees of freedom.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
LLF	Log likelihood.
RSQ	R-squared.
RBARSQ	RBAR-squared.
VCOV	Parameter covariance matrix.

Remarks

The PLS command carries out partial least squares. This algorithm chooses successive orthogonal factors from the independent variables that maximize the covariance between each X-score and the corresponding Y-score. Typically, the first few factors exhibit a high correlation, which then decreases from one factor to the next.

PLS is especially appropriate in the context of very many predictor variables relative to the number of observations, and can be used for finding a few underlying predictors that account for most of the variation in the response.

The structure of the equation to be estimated can be specified either by using a list of variables, with the dependent variable first, as in example 1 below, or by using an equation name which has been previously specified in a Type I FRML command. The maximum number of factors to be used is specified in *maxfactor* however the actual number used is determined when the percentage response variation that is explained by the current PLS component falls below *tolerance*.

Print options include p — pause after each screen display, d — print descriptive statistics, e — print elasticities, q — quiet - no screen or printed output, and s — print diagnostic statistics.

The diagnostic statistic option is used to generate cross validated Predicted REsidual Sum of Squares (PRESS) statistics. Each predicted residual is derived based on a *leave one out* jackknife estimation. The number of PLS factors (or components) is specified in *maxpress*. The number of PLS components to be used in a model can be based on a minimum PRESS statistic, or the number of components below which the reduction in the PRESS is insignificant.

The variables described in "Outputs" are returned as global variables; they can subsequently be used in any GAUSS or GAUSSX command.

See the "General Notes for Linear Models" under OLS, and the example given in test48.prg.

Example

```
1    LIST xlist;
         SYMBOL = x;
         RANGE = 1 15;
     PLS (p) y xlist ;
         ORDER = 5;

2.   FRML eq1 y c x1 x2 x3 x4 x5 x6 x7 x8;
     PLS (p,d,s) eq1;
```

```
ORDER = 4;
TOL = 0;
MAXIT = 7;
```

In example 1, a PLS is carried out with `y` as the dependent variable, and the fifteen `x` variables specified in `xlist`, with a maximum of five factors.

In example 2, PLS is performed on the structural equation specified in `eq1`. The constant is ignored. 4 factors are used, since *tolerance* is specified as zero. A table of PRESS statistics is generated (since (s) was specified as a print option) with up to seven factors.

See Also FRML, OLS, TITLE, WEIGHT

References

de Jong, S. (1993), "SIMPLS: An Alternative Approach to Partial Least Squares Regression", *Chemometrics and Intelligent Laboratory Systems*, Vol. 18, pp. 251-263.

Wold, H. (1966), "Estimation of Principal Components and Related Models by Iterative Least Squares", in *Multivariate Analysis*, ed. P. R. Krishnaiah, New York: Academic Press, pp. 391-420.

Purpose

Estimates the coefficients of a linear model where the dependent variable is drawn from a Poisson distribution.

Format

POISSON (*options*) *vlist ;*
MAXIT = *maxit;*
TITLE = *title;*
TOL = *tolerance;*
VALUE = *values;*
WEIGHT = *wtname;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
maxit	numeric, optional, maximum number of iterations (20).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
values	numeric, optional, starting value of coefficients.
wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of T–statistics.
LLF	Log likelihood.
VCOV	Parameter covariance matrix.

Remarks

The POISSON command estimates the parameters of a linear model in which the dependent variable is drawn from a Poisson distribution, with parameter λ_i, which is related to the regressors, x_i. The model is:

$$Prob(Y_i = y_i) = \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}, \qquad y_i = 0, 1, 2, ...$$

where λ_i is given by:

$$\ln \lambda_i = \beta' x_i.$$

GAUSSX uses the current sample to estimate the Poisson process, automatically dropping missing values. The dependent variable must be integer. Print options include (p) pause, and (i) display parameters at each iteration.

Estimation takes place using Newton's method. Convergence is not guaranteed. Starting values for the structural component are estimated using OLS in the default, but can be explicitly given using the VALUE option.

See the "General Notes for Linear Models" under OLS, and an example is given in test09.prg. Non-linear models can be estimated using maximum likelihood - see POISSON process.

Example

```
FRML eq1 y c x1 x2;

1.   POISSON eq1 ;

2.   POISSON (p,i,d) y c x1 x2 x3;
        MAXIT = 40;
        VALUE  = 1 .2 .1 0;
```

In example 1, a Poisson process is modelled based on the equation eq1; starting values for the regression are based on OLS.

In the second example, the starting values for the independent variables (c, x1. x2, x3) are given using the VALUE option. 40 iterations are permitted. Descriptive statistics (d) are printed, the display pauses (p) at each screen, and parameter values are shown at each iteration (i).

See Also FRML, NLS, OLS, POISSON, TITLE, WEIGHT

References Greene, W.H. (1993), *Econometric Analysis*, 2nd ed. Macmillan, New York.

Purpose
Creates a vector of log likelihoods for a Poisson process.

Format
z = POISSON (*y, indx, trunc*);

Input

y	literal, dependent variable.
x	literal, index of independent variables.
trunc	literal, truncation vector.

Output

z	Vector of log likelihoods.

Remarks
The Poisson coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models. *trunc* is a two element vector consisting of the lower and upper truncation points, or 0 for no truncation.

For details of the Poisson model see POISSON. Also see the "General Notes for Non-Linear Models" under NLS. An example is given in test09.prg.

Example

```
OLS y c x1 x2;
PARAM a0 a1 a2;
   VALUE = coeff;
FRML eq1 indx = a0 + a1*x1 + a2*x2;
FRML eq2 lf = poisson(y,indx,0);
ML (p,d,i) eq1 eq2;
```

In this example, a standard linear Poisson model is estimated, using OLS starting values. The RHS index is stipulated in eq1, and the log likelihood is returned from eq2.

Source
GXPROCS.SRC

See Also
ML, NLS, POISSON

Purpose

To compute the principal components for a given set of vectors.

Format

PRIN (*options*) *plist;*
VLIST = *vlist;*

Input

options	optional, print options.
plist	literal, required, principal component list.
vlist	literal, required, variable list of original series.

Output

_MEANS	Vector of means.
_STDS	Vector of standard deviations.
_FACTOR	Factor matrix.

Remarks

PRIN computes the principal components of the list of variables given in *vlist* under the present sample, and stores them in *plist*. These variables are then stored in the GAUSSX workspace. Missing values are listwise excluded.

The original series are first standardized to have zero mean and unit variance. After estimating the eigen values and vectors of the X'X matrix, the standardized variables are post-multiplied by the factor loadings to create the principal components. These will also be standardized, so as to have zero mean, unit variance, and orthogonal.

The print options include d — display the characteristic roots and factor loading matrix, c — print the correlation matrix for the original variables, and p — pause after each screen display. On-line help is also available.

If the number of elements in *plist* is less than in *vlist*, the set of principal components will explain less that 100% of the variance of the original variables. If one of the original variables is a constant, there will be a zero characteristic root, and one of the principal components will be a vector of zeros.

Example

```
PRIN (p,d,c) p1 p2 p3 ;
   VLIST = c gnp inv ;
```

In this example, three principal components are created – p1, p2, and p3 from the series c, gnp, and inv. A correlation matrix is displayed under the (c)

option, and the factor loadings and characteristic roots are displayed under the (d) option. Since c is a vector of unity, one of the roots is zero, and p3 will be a vector of zeros.

See Also DIVISIA, SAMA

References Judge, G.G. *et. al.* (1985), *The Theory and Practice of Econometrics*, John Wiley & Sons, New York.

Purpose

Prints vectors.

Format

PRINT (*options*) *vlist ;*
FMTLIST = *fmtopts;*
RANGE = *rangelist;*

Input

options	optional, print options.
vlist	literal, required, variable list.
fmtopts	literal, optional, format options.
rangelist	literal, optional, range list.

Remarks

The PRINT command prints out the data for the specified variables, under the current sample. The default number of lines per page is 40; this can be changed using MAXLINES argument in the OPTION command. A subset of the current sample can be printed out, without creating a new sample file, using the RANGE option. Formatting is available using the FMTLIST option.

Example

```
SMPL 1968 1977;

1.   PRINT y x1 x2;

2.   PRINT (p) y x1 x2;
        RANGE = 1970 1974;

3.   PRINT x1 x1(-1);
```

In the first example, ten observations for each of the variables is printed out. In the second, only 5 observations are printed, and execution pauses (p) after each screen. In the third example, x1 and x1 lagged once are printed.

See Also

COVA, FMTLIST

Purpose
Creates a vector of log likelihoods for a multivariate binomial Probit process.

Format
z = PROBIT (*ymat, xmat, rvec*);

Input

ymat	literal, matrix of alternative chosen.
xmat	literal, matrix of utility values for each alternative.
rvec	literal, vector, correlation coefficients.

Output

z	Vector of log likelihoods.

Remarks
The structural and correlation coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models. Models include univariate, bivariate and trivariate probit.

An example is given in test40.prg.

Example

```
FRML eq1 xb1 = a0 + a1*x1 + a2*x2;
FRML eq2 xb2 = b0 + b1*x3 + b2*x4;
FRML ellf llf = probit(y1~y2,xb1~xb2,r12);
FRML ec1 r12^2 <= .9999;
ML (p,i) eq1 eq2 ellf;
   METHOD = bhhh bhhh nr;
   EQCON = ec1 ;
```

This example estimates a bivariate probit model, restricting the correlation coefficient to lie in the prescribed range.

Source
PROBITX.SRC

See Also
ML, NLS, PDROOT, QR

Purpose

Saves a matrix to an ASCII or binary file.

Format

ret = PUTM (*filename, x, mode, append*);

Input

filename	string, name of output file.
x	*N*x*K* matrix to be written to filename.
mode	scalar, file mode, (0) binary or (1) ASCII.
append	scalar, file write mode, (0) overwrite or (1) append.

Output

ret scalar, return code

Remarks

PUTM writes a matrix to an ASCII or to a binary data file of type double.

The return code, *ret*, takes the following values:

0	normal return
1	null file name
2	file open error
3	file write error
4	illegal append value

PUTM is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx;
ret = putm(c:\temp\mydata.bin,x,0,0);
```

This example writes the matrix `x` as binary to c:\temp\mydata.bin. If the file currently exists, it will be overwritten.

Source

GXPROCS.SRC

See Also

GETM

Purpose
Calculate the present value of a stream of payments.

Format
y = PV (*pmt, r, nper*);

Input

pmt	*N*x1 vector, or scalar, periodic payment.
r	*N*x1 vector, or scalar, interest rate at each period.
nper	scalar, number of periods.

Output

y	Scalar, present value of the periodic payments.

Remarks
The PV statement returns the present value of a stream of payments over time. The payment is made at the end of each period; thus the first term of the series is *pmt*[1]/(1+*r*). If *pmt* is a scalar, then the payment stream consists of *pmt* at each period. If *r* is a scalar, then the discount rate is assumed the same over the *nper* periods. If *pmt* and/or *r* are vectors, they must have lengths of *nper*. Interest rate is per period; thus an annual rate of 9% paid monthly for 20 years would have$r = .09/12 = 0.0075$, and $n = 12 * 20 = 240$.

PV is pure GAUSS code, and is used independently of GAUSSX.

Example

```
library gaussx ;
pmt  = 100;
r    = .1/12;
nper = 120;
pval = pv(pmt,r,nper);

pval = 7567.116
```

This calculates the present value of a stream of payments of $100 per month for 10 years, with a discount rate of 10%

Source
FINANCE.SRC

See Also
AMORT, FV, MCALC

Purpose

Integrates the *K*-variate normal density function over a range of upper and lower bounds.

Format

y = QDFN (*xh*, *xl*, *omega*);

Input

xh *K*x1 or *K*x*N* matrix, the upper limits of the *K*-variate normal density function.

xl *K*x1 or *K*x*N* matrix, the lower limits of the *K*-variate normal density function.

omega *K*x*K* symmetric, positive definite covariance matrix of the *K*-variate normal density function for the exact or simulation case. *K*x(*R* + 1) for the factor analytic case, where the covariance matrix has *R* factors.

_qdfmth global scalar, the choice of method
_qdfmth = 0. The normal density function is evaluated using internal GAUSS functions if $K \leq 3$.
_qdfmth = 1. The probability is evaluated using a smooth recursive simulator. *K* is unrestricted.
_qdfmth = 2. The probability is evaluated using a factor analytic method providing the covariance matrix has three or less factors. *K* is unrestricted.

_qdfrep global scalar, the number of replications. (20)

_qdfrlz global scalar, the number of realizations. (1)

_qdford global scalar, the order of the integration:
2, 3, 4, 6, 8, 12, 16, 20, 24, 32, 40. (16)

Output

y *N*x1 vector of the estimated integrals evaluated between the limits given by *xh* and *xl*.

Remarks

This procedure returns the probability of an *K*-dimensional rectangle where the probability density function is *K*-dimensional normal. It can evaluate the probability for a single set of upper and lower bounds, or can evaluate the probability for *N* points, providing the covariance matrix is constant.

The evaluation in the case when *K* is three or less can be carried out using Gauss functions. For *K* greater than 3, two methods are employed. The first

is a factor analytic. This is only feasible if the covariance matrix has a limited factor structure. That is, the covariance matrix (Ω) can be written as:

$$\Omega = D + BB'$$

where D is a KxK diagonal matrix, and B is a KxR matrix, where R is the number of factors. R cannot be greater than 3. For example, for the single factor case, B will be a Kx1 vector. The factor analytic method is exact for those cases where the covariance matrix satisfies the relationship shown above, providing _qdford is set sufficiently high – the default is 16, but higher values may be necessary for the $R = 3$ case.

The second method that can be used for large K is the method of simulation. Until recently, these methods have converged too slowly to be of sufficient interest. Recent work has resulted in a number of simulators being proposed that are consistent, and converge very quickly. The most promising is the smooth recursive simulator proposed by Geweke, Hajivassiliou and Keane (GHK). It is far slower than factor analytic for the same degree of accuracy, but can be used for any positive definite covariance matrix. The accuracy can be increased by increasing either _qdfrep, the number of replications, and/or _qdfrlz, the number of realizations.

QDFN is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
let xh[3,2] = 1  1 2  0  3  1 ;
let xl[3,2] = 0 -2 0 -5  0 -2 ;
let b[3,2] = .3 .1 .5 -.3 .7 .6;
let d = 1 1.5 2 ;
omega = eye(3).*d + b*b';
vmat = d~b ;
_qdfmth = 0;
z0 = qdfn(xh,xl,omega)
_qdfmth = 1;
z1 = qdfn(xh,xl,omega)
_qdfmth = 2;
z2 = qdfn(xh,xl,vmat)
z = z0~z1~z2; z;
```

```
          0 -2            1   1
    xl =  0 -5      xh =  2   0
          0 -2            3   1

          0.3  0.1        1.0
     b =  0.5 -0.3    d =  1.5
          0.7  0.6        2.0

                1.10  0.12  0.27            1.0  0.3  0.1
    omega =     0.12  1.84  0.17      vmat = 1.5  0.5 -0.3
                0.27  0.17  2.85            2.0  0.7  0.6

z =       0.072139    0.071460    0.072139
          0.251917    0.252609    0.251917
```

This integrates the two factor 3-variate normal density function over the specified range for two observations.

Source QDFN.SRC

See Also CDFN, CDFBVN, CDFTVN, CDFMVN, INTQUAD, INTQUAD2, INTQUAD3

Purpose

Estimate the coefficients of a linear model with a qualitative dependent variable (Quantal Response), using binomial probit, multinomial logit, and ordered logit and probit.

Format

QR (*options*) *elist ;*
CATNAME = *categories;*
MAXIT = *maxit;*
METHOD = *methname;*
TITLE = *title;*
TOL = *tolerance;*
WEIGHT = *wtname;*

Input

options	optional, print options.
elist	literal, required, variable list or equation name.
categories	literal, optional, list of category names.
maxit	numeric, optional, maximum number of iterations (20).
methname	literal, optional, algorithm name (LOGIT).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
wtname	literal, optional, weighting variable.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
DPDX_B	Vector of marginal effects.
DPDX_SE	Vector of std. error of marginal effects.
DPDX_T	Vector of t-stat. of marginal effects.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error elasticities.
ETA_T	Vector of t-stat. of elasticities.
LLF	Log likelihood.
VCOV	Parameter covariance matrix.
MEANS	Means of the independent variables.
STDS	Standard deviations of the independent variables.
PERCNT	Percent cases in each outcome category.

Remarks

GAUSSX will also recognize the commands LOGIT, PROBIT, ORDLGT, and ORDPRBT. This type of model requires that the dependent variable be qualitative in nature, and thus takes only g consecutive values corresponding to the g categories. A constant (C) need not be placed in the equation – GAUSSX adds the required number of constants automatically.

The available methods are:

[LOGIT]	Multinomial LOGIT.
PROBIT	Binomial PROBIT.
ORDERED LOGIT	Ordered LOGIT.
ORDERED PROBIT	Ordered PROBIT.

In a multinomial estimation, each additional category results in a set of additional coefficients; thus if there are k explanatory variables, and g categories, there will be $(g-1)\times(k+1)$ coefficient values. Non-linear multinomial logit can be carried out using MNL. For an ordered logit (or probit), the coefficient values remain fixed, and only the constant changes; thus there will be $k+(g-1)$ coefficient values.

For the binomial probit, only two categories can be specified. Multinomial probit and non-linear probit can be carried out using MNP.

The coefficients in the LOGIT and PROBIT estimation procedures are defined so that their sign conforms to the "industry standard" - a positive coefficient implies a higher propensity to be in the selected group. The reference category is set for the first category.

The marginal effect – $\partial P/\partial X$ – and their associated standard errors and t-statistics are also reported for all QR models, evaluated at the sample mean. These are stored as globals under the names DPDX_B, DPDX_SE, and DPDX_T respectively. Alternatively, elasticities are also available; these are stored as globals under the names ETA_B, ETA_SE, and ETA_T respectively.

Print options include p—pause after each screen display, d — print descriptive statistics, e — print elasticities, i — print parameters at each iteration, m — print marginal effects, and q — quiet - no screen or printed output.

A sample file containing just the cases and variables relevant for the current estimation is created prior to the estimation. Missing values are listwise deleted from this subset of variables. See "General notes for Non-Linear Estimations" for details.

The FORCST command works as expected; for example, if example 2 had just been estimated, then the statement:

```
FORCST prblue prcraft prwhite prprof;
```

will generate four variables corresponding to the predicted probability of a case occurring in each category. Note that the number of arguments in FORCST equals the number of categories. The FORCST command can also be used to compute Mill's ratio after a PROBIT or LOGIT estimation. The LOGIT Mill's ratio is defined using the Trost and Lee (1984) methodology. Thus a polychotomous choice model can be estimated using LOGIT, and OLS applied to each category, correcting for selectivity. The number of Mills variables must equal the number of categories – see example 5 below and test08.prg. Sample selection models can also be run using the HECKIT command.

See the "General Notes for Linear Models" under OLS, and the examples given in test08.prg.

Example

```
FRML eq1 y x1 x2 x3;

1.   QR  y1 x1 x2 x3;
        METHOD = probit;
        CATNAME = Blue White;

2.   QR (d,p) eq1;
        METHOD = logit;
        CATNAME = Blue Craft White Prof;

3.   QR eq1;
        METHOD = ordered logit;
        MAXIT = 10;
```

```
4.   QR (i,p) y x1 x2 x3;
        METHOD = ordered probit;
        CATNAME = Blue Craft White Prof;

5.   PROBIT eq1;
     FORCST mr;
        MODE = mills;
     SMPL y;
     OLS wage c z1 z2 z3 mr;
        METHOD = robust;
```

In the first example, a binomial probit is estimated - y1 takes one of two values (eg zero or unity); the explanatory variables are x1, x2 and x3. The user can specify names for each category by using the CATNAME option.

In example 2, a multinomial logit is estimated on four categories; in this case the equation is specified as the name of a previously defined FRML. Descriptive statistics (d) are produced, and execution pauses (p) after each screen display. The dependant variable (y) would take the value 1, 2, 3, or 4.

Example 3 repeats the previous example, but uses an ordered logit, as opposed to a multinomial logit. A maximum of 10 iterations is specified.

In example 4, an ordered probit is carried out on four categories. The iteration (i) option generates detailed information for each iteration, and pauses (p) after each display,

Equation 5 shows how a two stage Heckman procedure for correcting selection bias can be carried out. A wage equation is to be estimated, but only employed persons (y = 1) have a wage. Estimate a probit on the entire sample, and generate Mills ratio mr. Then on the sample of working individuals, carry out an OLS including mr as an explanatory variable (and correcting for heteroscedasticity by using ROBUST).

See Also FRML, HECKIT, MNL, MNP, OLS, TITLE, WEIGHT

References

Amemiya, T. (1981), "Qualitative Response Models: A Survey", *Journal of Economic Literature*, Vol. 19, pp. 1483-1536.

Bera, A.K., C.M. Jarque, and L.F. Lee. (1984). "Testing the normality assumptions in limited dependent variable models", *International Economic Review*, Vol. 25(3), pp 563-578.

Maddala, G.S. (1983), *Limited-dependent and Qualitative Variables in Econometrics*, Cambridge University Press, Cambridge.

Trost, R.P., and L.F. Lee (1984), "Technical Training and Earnings: A Polychotomous Choice Model with Selectivity", *Review of Economics and Statistics*, Vol. 66(1), pp. 151-156.

Purpose To change the name of a GAUSSX variable.

Format RENAME *oldname newname ;*

Input
oldname literal, required, existing variable name.
newname literal, required, new variable name.

Remarks The RENAME statement is much faster than a GENR followed by a DROP, since only the header of the GAUSSX data file is changed. RENAME is not applicable on UNIX platforms.

Example

```
RENAME gnp newgnp;
```

The variable gnp is renamed newgnp.

See Also DROP, KEEP, STORE

Purpose

Creates a matrix of (pseudo) random variables derived from the specified distribution.

Format

y = RND (*pdfname*, *r*, *c*, *p1*, *p2*, *p3*);

pdfname	string, the name of the probability distribution.
r	scalar, the row dimension.
c	scalar, the column dimension.
p1	*R*x*C* matrix or scalar, first parameter for the specified distribution.
p2	*R*x*C* matrix or scalar, second parameter for the specified distribution.
p3	*R*x*C* matrix or scalar, third parameter for the specified

Output

y	*R*x*C* matrix of random variates.

Remarks

This procedure returns pseudo random variates from the specified distribution.

See the "General Notes for Probability Density Functions" under PDF.

RND is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
a = 2; ;
y = rnd(poisson,2,3,a,0,0);

  y =  1.0000    4.0000    2.0000
       3.0000    1.0000    2.0000
```

This generates a random sample of 6 observations from a Poisson distribution with $\lambda = 2$.

Source

PDFX.SRC

See Also

CDF, CDFI, PDF, RND, RNDGEN, RNDTN

Purpose

Creates a matrix of (pseudo) random variables derived from a specified cumulative distribution function.

Format

y = RNDGEN (*&cdf, r, c, dta, pvec, cns*);

&cdf	pointer to a function cdf(x,dta,pvec), defined as a procedure.
r	scalar, the row dimension.
c	scalar, the column dimension.
dta	optional data matrix used by the cdf procedure, or zero.
pvec	*k*x1 parameter vector used by the cdf procedure
cns	scalar, argument constraints.

Output

y	*r*x*c* matrix of random variates.

Remarks

This procedure returns pseudo random variates for the specified distribution. The cumulative distribution function for the required distribution is specified as *cdf(x, dta, pvec)*, where *x* is the argument, *dta* is an optional data matrix, and *pvec* is an optional parameter vector. If each row of *dta* is to be used for each invocation, then *dta* must have *r* rows.

The permitted range of the argument is specified as a scalar in *cns*. The available values are:

0	$-\infty < x < \infty$.
1	$x \geq 0$.
2	$x \leq 0$.
3	$0 \leq x \leq 1$.

RNDGEN uses Newton's method, and so convergence is not guaranteed. If convergence fails, a missing value is returned.

RNDGEN is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;

proc beta_cdf(x,dta,pvec);
  local v1, v2, xx, cdf;
  v1=pvec[1];
  v2=pvec[2];
  cdf = cdfbeta(x,v1,v2);
  retp(cdf);
endp;

pvec = { .3, .5 };
cns = 3;
y = rndgen(\&beta_cdf,100,1,0,pvec,cns);
```

This generates a random sample of 100 observations from a beta distribution with shape parameters .3 and .5. Since the argument for a beta variate lies in the range {0:1}, *cns* is specified as 3.

Source GXPROCS.SRC

See Also RND

Purpose Creates a matrix of quasi random variables.

Format *y* = RNDQRS (*n*, *k*);

Input
n Row dimension.
k Column dimension (max 6).

Output
y *N*x*K* matrix of quasi random sequence.

Remarks The Sobol sequence generator generates quasi random sequences of numbers up to 6 dimensions. Such sequences fill a space more uniformly than uncorrelated random sequences - in a sense they are maximally avoiding of each other. (Press *et. al.* (1993), p. 300).

RNDQRS is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
y = rndqrs(1000,4);
```

This example generates a 1000 by 4 matrix y of quasi random sequences.

Source BITWISE.SRC

See Also RND, RNDSMPL

References Press, W.H. *et. al.* (1993), *Numerical Recipes*, Cambridge University Press, New York.

Purpose Random sampling from a population with or without replacement.

Format *y* = RNDSMPL (*m*, *n*, *c*);

Input

m	Number of elements required in sample.
n	Number of elements in population.
c	Flag - No replacement if *c* = 0, replacement if *c* = 1.

Output

y	*m*x1 vector of integers between 1 and *n*.

Remarks If the flag *c* is set to zero, *y* will consist of unique numbers between 1 and *n*; consequently, *m* <= *n*. If *c* is set to unity, replacements are permitted. The output *y* can then be used as an index for the sample.

RNDSMPL is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
y = rndsmpl(50,100,0);
```

This example generates a 50 element vector y of integers between 1 and 100, with no replacements.

Source GXPROCS.SRC

Purpose

Creates a matrix of (pseudo) random variables distributed truncated multivariate normal.

Format

y = RNDTN (*xh, xl, mu, omega*);

Input

xh *K*x1 or *K*x*N* matrix, the upper limits of the *K*-variate normal density function.

xl *K*x1 or *K*x*N* matrix, the lower limits of the *K*-variate normal density function.

mu *K*x1 or *K*x*N* matrix, means of the *K*-variate normal density function.

omega *K*x*K* symmetric, positive definite covariance matrix of the K-variate normal density function.

_rtnrep global scalar, the number of Gibbs replications (default = 20).

_rtnpnt global scalar, 1 - print iteration number (default = 0).

Output

y 1x*K* vector or *N*x*K* matrix of random numbers derived from the multivariate normal density function between the limits given by *xh* and *xl*.

Remarks

The truncated multivariate normal number generator is based upon the uniform number generator, and uses the same seed. The methodology uses the Gibbs Sampler, which is based on a Markov chain that utilizes univariate truncated normal densities to construct conditional variates, and has the truncated multivariate normal as its limiting distribution.

RNDTN is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx ;
let xh = 2 1;
let xl = 0 -1;
let omega[2,2] = 1 .8 .8 1;
let mu[2,5] = 3  3  3  0  0
              0  0  0  0  0;
y = rndtn(xh,xl,mu,omega);
```

```
 xh =  2       xl =  0    mu =     3  3  3  0  0
       1             -1            0  0  0  0  0


 omega =  1.0   0.8
          0.8   1.0


  y =    1.7296491       0.02386495
         1.9555895       0.14874314
         1.9291931      -0.42081598
        0.79121303       0.56267756
        0.34698671      -0.51624094
```

This simulates the bivariate truncated normal density function over the specified delineation range for five observations with specified means.

Source RNDTN.SRC

See Also RND, RNDN, RNDU

References

Hajivassiliou, V. (1992), "Simulation Estimation Methods for Limited Dependent Variable Models" in *Handbook of Statistics*, Vol. 11 (Econometrics), G.S. Maddala, C.R. Rao, and H.D. Vinod (eds.). Amsterdam: North Holland.

Purpose

Estimates the coefficients of a linear equation using robust estimation.

Format

ROBUST (*options*) *vlist ;*
MAXIT = *maxit;*
METHOD = *methname;*
PDL = *pdllist;*
REPLIC = *replic;*
TITLE = *title;*
TOL = *tolerance;*
VALUE = *value;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
maxit	numeric, optional, max. number of iterations (20).
methname	literal, optional, estimation method (LAD)
pdllist	literal, optional, options for PDL.
replic	numeric, optional, replication options.
title	string, optional, title.
value	numeric, required, estimation parameter.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
DF	Degrees of freedom.
RSS	Residual sum of squares.
SER	Standard error of the regression.
SAR	Sum of absolute residuals.
LLF	Log likelihood.
RSQ	R-squared.
RBARSQ	RBAR-squared.
VCOV	Parameter covariance matrix.

Remarks

The ROBUST command carries out robust estimation of linear equation models. A number of different estimation methods are available; in each case, the objective function depends on the absolute value of the residuals. This type of regression will generally be used when the disturbance distribution is unknown; in these circumstances, ROBUST provides better estimates of the underlying parameters than OLS, especially when the residuals have fat tails (leptokurtic).

GAUSSX provides six robust estimation methods to evaluate β in the linear equation model:

$$y_t - x_t'\beta + \epsilon_t$$

Quantile Regression is an L-estimator; it is carried out using the interior point algorithm of Park and Koenker. β is obtained from the solution of:

$$\min_{\beta}\left[\sum_{\{t\,|\,y_t \geq x_t'\beta\}} \rho\ \left|y_t - x_t'\beta\right| + \sum_{\{t\,|\,y_t \leq x_t'\beta\}} (1-\rho)\ \left|y_t - x_t'\beta\right|\right]$$

When $\rho = .5$, quantile regression is equivalent to LAD.

The other five methods are M-estimators, and are evaluated using iterated re-weighted least squares, with weights W_t:

Least Absolute Deviation:

$$W_t = \left(\frac{1}{\left|y_t - x_t'\beta\right|}\right)$$

Huber's t Function:

$$W_t = 1 + \sup_{|y_t - x_t'\beta| > \rho}\left(\frac{\rho}{\left|y_t - x_t'\beta\right|} - 1\right)$$

Ramsay's E Function:

$$W_t = \exp(-\rho\ \left|y_t - x_t'\beta\right|)$$

Andrew's Wave Function:

$$W_t = \sup_{|y_t - x_t'\beta| < \pi\rho}\left(\frac{\sin\left[(y_t - x_t'\beta)/\rho\right]}{(y_t - x_t'\beta)/\rho}\right)$$

Tukey's Biweight:

$$W_t = \sup_{|y_t - x_t'\beta| < \rho} \left(1 - [(y_t - x_t'\beta)/\rho]^2\right)^2$$

Specification The structure of the equation to be estimated can be specified either by using a list of variables, with the dependent variable first, as in example 1 below, or by using an equation name which has been previously specified in a Type I FRML command.

Estimation Method The estimation method used is specified in *methname*. The available methods are:

QR	Quantile Regression.
LAD	Least Absolute Deviation.
HUBER	Huber's t Function.
RAMSAY	Ramsay's E Function.
ANDREW	Andrew's Wave Function
TUKEY	Tukey's Biweight Function

The parameter, ρ, is specified in *value*.

Convergence Both the interior point algorithm and the iterated reweighted least squares involve iterations until convergence. Convergence is declared when the proportional change in each parameter is less than *tolerance* the default is 0.001. If *tolerance* consists of two elements, the first element represents the maximum proportional change in each parameter for convergence, and the second element represents the maximum proportional change in the objective function for convergence – convergence is achieved when either of these criteria is achieved. If convergence is not achieved within *maxit* iterations, estimation is terminated, and the current parameter results are displayed.

Lags Lagged variables can be used by specifying the lag in parenthesis - see example 2. Polynomial distributed lags can be specified using the PDL option.

Inference In most cases, the distribution of the residuals in the estimation equation is unknown - indeed, this is why one is using a robust estimate.

Consequently, the distribution of the coefficient estimates is unknown. An estimate of the parameter covariance matrix can be derived using bootstrapping. The bootstrap procedure is controlled by *replic*:

```
REPLIC = num nsize npnt;
```

where `num` is the number of bootstrap replications on the data that has been drawn `nsize` times, with replacement, from the current sample. The iteration count is printed out every `npnt` iterations. The default (and recommended) value for `nsize` is the current sample size.

The mean bootstrap coefficient values and the 95% percentile band are displayed, and the parameter covariance matrix is derived from the bootstrap data. A bootstrap is not carried out if `REPLIC` is not specified, and the covariance matrix is set to missing.

Regression Statistics The regression statistics for ROBUST are derived using standard formulae, except that standard errors and t-stats are derived from the bootstrap estimate of the parameter covariance matrix, and the log likelihood is evaluated assuming that the residuals are distributed with a Laplace distribution. Measures of fit are not constrained in the same way as in OLS; thus, for example, R-squared can easily be negative.

Print Options These include p — pause after each screen display, b — brief output, d — print descriptive statistics, i — print parameters at each iteration, q — quiet - no screen or printed output, and s — print diagnostic statistics.

An example of ROBUST is given in test32.prg.

Example

```
1.   ROBUST y c x1 x2 ;

2.   ROBUST (d,p,s) y c x1 x2(-1);
        METHOD = qr;
        REPLIC = 100;
        VALUE = .25;

3.   FRML eq1 y c x1 x2;
     ROBUST (d, i, p, s) eq1;
```

```
METHOD = huber;
REPLIC = 100;
VALUE = .4;
PDL = x1 2 4 none;
MAXIT = 100;
```

In example 1, a ROBUST estimation is carried out with y as the dependent variable, and c, x1 and x2 as the independent variables. The LAD method is used as the default, and no bootstrap is carried out.

In example 2, a quantile regression is estimated (with $\rho = .25$), with the same variables as in example 1, but with x2 replaced with its lagged value. The parameter covariance matrix is estimated based on 100 bootstrap replications. In this case descriptive statistics (d) and diagnostic statistics are produced (s), and execution pauses (p) after each screen display.

In example 3, a robust estimation using Huber's t function with $\rho = .4$ is performed on the structural equation specified in eq1, but with a PDL estimation occurring for x1.

See Also FRML, OLS, PDL

References

Judge, G.R., *et. al* (1988), *Introduction to the Theory and Practice of Econometrics*, 2nd ed., John Wiley, New York.

Koenker, R.W. and G.W. Bassett (1978), "Regression Quantiles", *Econometrica*, Vol 46, pp. 33-50.

Portnoy, S., and R. Koenker (1997), "The Gaussian Hare and the Laplacean Tortoise: Computability of Squared-error vs. Absolute Error", *Statistical Science*, Vol. 12.

Staudte, R.G. and S.J. Sheather (1990), *Robust Estimation and Testing*, John Wiley, New York.

Purpose

Estimates the optimum factors for a response surface problem.

Format

RSM (*options*) *elist;*
EQCON = *cnstrntlist;*
EQSUB = *macrolist;*
GENALG = *genalg;*
GLOBOPT = *globopt;*
MAXIT = *maxit;*
MAXSQZ = *maxsqz;*
METHOD = *methname;*
MODE = *metric;*
POSDEF = *pdname;*
SIMANN = *simann;*
TITLE = *title;*
TOL = *tolerance;*
USERPROC = *&userproc;*
VALUE = *value;*
VLIST = *matname;*

Input

options	optional, print options.
elist	literal, required, equation list.
cnstrntlist	literal, optional, list of constraint equations.
macrolist	literal, optional, macro equation list.
genalg	numeric, optional, GA options (`30,4 .4 .25`).
globopt	numeric, optional, GO options (`20000 100 .0001 4`).
maxit	numeric, optional, maximum number of iterations (20).
maxsqz	numeric, optional, maximum number of squeezes (10).
methname	literal, optional, algorithm list (`BFGS GA NR`).
metric	literal, optional, desirability algorithm or metric (`DS`).
pdname	literal, optional, positive definite algorithm (`NG`).
simann	numeric, optional, SA options (`5 .85 100 20`).
title	string, optional, title.
tolerance	numeric, optional, param. convergence tolerance (.001).
&*userproc*	literal, optional, pointer to user specified desirability procedure.
value	literal, optional, metric parameter or matrix.
matname	literal, required, response parameter bound matrix.

Values in parentheses are the default values.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
LLF	Model criteria.
VCOV	Parameter covariance matrix.
_RSMRSPN	Vector of responses.
_RSMDSR	Vector of desirabilities

Remarks

RSM is a methodology that is used extensively in engineering to solve an optimization problem using simulation. In the first step, an experimental design is used to fit a set of observed responses (r) to a set of factors; typically this can be modelled by NLS using a non-linear model, or by OLS or SURE on a polynomial expansion of the factors (see XPAND). A desirability measure or distance metric (generalized distance) is specified as a function of the (fitted) responses, and in the second step the optimal factor choice that maximizes the desirability measure (or minimizes the distance metric) is determined by optimization. Upper (r_u), lower (r_l) and target (r_t) values of the responses may be specified, conditional on the metric chosen. Since the response surface is not smooth, and typically has many local optima, the genetic algorithm is used as the default.

The RSM command estimates the parameters of a model via the maximum likelihood method - in effect, the desirability is used instead of the likelihood. The user specifies FRMLs which compute the response variables given a set of factors. For each iteration, the computed response variables are then used as an argument to the metric specified in *mode* (or &*userproc*). The form is very similar to ML.

Four estimation methods are generally available: BFGS, DFP, BHHH, and NR. In addition, since RSM models typically have many local optima, one can use the genetic algorithm (GA), simulated annealing (SIMANN), global optimization (GO) or the Nelder-Meade algorithm (NM) for the second element of METHOD - the default is GA.

The metric is specified in *mode*, or a user specified desirability function can be programmed in &*userproc*. The available algorithms are:

DS	Derringer and Suich desirability function (default).
HAR	Harrington's desirability function.
EUCLID	Euclidian distance. $d^2 = (r - r_t)'(r - r_t)$. This is best suited where the responses are similar.
STD	Standardized Euclidian distance. $d^2 = (r-r_t)'D^{-1}(r-r_t)$ where D is the diagonal matrix of the variance of the responses used in the design stage. The vector of the standard deviations of the design responses is specified in *value*. This is recommended where the responses have different variances.
MAHAL	Mahalanobis distance. $d^2 = (r - r_t)'V^{-1}(r - r_t)$ where V is the covariance matrix of the responses used in the design stage. The design responses covariance matrix is specified in *value*. This is recommended where the responses have different variances, and are correlated.
CHEB	Chebyshev metric. $d = \max_{i=1,\ldots,n} \mid (r_i - r_{t,i}) \mid$.
CITY	City Block (or Manhattan) metric. $d = \sum_{i=1}^{n} \mid (r_i - t_{t,i}) \mid$.
MINK	Minkowski metric. $d = \left(\sum_{i=1}^{n} \mid r_i - r_{t,i} \mid^{\rho}\right)^{1/\rho}$. The value of ρ is specified in *value*.

Each desirability measure or distance metric requires parameters qualifying each response. Thus the user needs to specify an *m*x6 matrix, where *m* is the number of responses. The elements of this matrix are given below:

Derringer and Suich

column 1	L - lower bound
column 2	T - desired or acceptable value
column 3	U - upper bound
column 4	index weight, $y \leq T$
column 5	index weight, $y \geq T$
column 6	response type: 'min' 'target' 'max'

Harrington

One Sided

column 1	L - lower value
column 2	U - upper value - $U > L$
column 3	assumed desirability for L
column 4	assumed desirability for U
column 5	weight
column 6	response type: 'one'

Two Sided

column 1	L - lower bound
column 2	T - desired value
column 3	U - upper bound
column 4	assumed desirability for T
column 5	weight
column 6	response type: 'two'

Distance Metrics

column 1	L - lower bound
column 2	T - target value
column 3	U - upper bound
column 4	weight
column 5	0
column 6	0

For the desirability models, see the references for details. For the distance metrics, the bounds provide optional limits on the deviations; they cannot exceed $r - r_l$ or $r_u - r$ respectively. The weights are applied multiplicatively to the individual deviations.

See the "General Notes for Non-Linear Models" under NLS, as well as ML. An example is given in test50.prg.

Example

```
FRML eq0  xmat := xpand(x1~x2~x3,2);
FRML eqy1 y1 = xmat*amat;
FRML eqy2 y2 = b0 + exp(b1*x1+b2*x2+b3*x3 + b4*x1^2);
let  pmat[2,6]  =    120  170   0 1 1 max
                     400  500 600 1 1 target;
PARAM amat;
  SYMBOL = a;
  ORDER = 10 1;
PARAM b0 b1 b2 b3 b4;
NLS (q) eqy1;
   EQSUB = eq0;
NLS (q) eqy2;
CONST amat;
CONST b0 b1 b2 b3 b4;
PARAN x1 x2 x3;
   VALUE =  0 0 0;
   LOWERB  = -1 -1 -1;
   UPPERB  =  1  1  1;
RSM (p,i)  eqy1 eqy2 ;
   EQSUB = eq0;
   MAXIT = 40;
   MODE = ds;
   VLIST = pmat;
```

This example demonstrates a 3 factor, 2 response, RSM example. The first equation is a polynomial expansion of the three factors to order 2 - this results in 10 terms, using XPAND. The second response has a specific non-linear form. The two responses are first estimated using NLS. The parameters of the estimation equation are then held as constants, and the factors (`x1, x2, x3`) now become the parameters. The second step involves the optimization of the desirability measure, in this case the default (Derringer and Suich). A matrix *pmat*, which defines the values required for the specified desirability measure, is specified in *vlist*. Optimization occurs in the RSM command similarly to ML using the genetic algorithm as the main method for locating the optimum.

Source RSMX.SRC

See Also FRML, ML, NLS, XPAND

References Derringer, G.C. and R. Suich (1980), "Simultaneous Optimization of Several Reasons Variables", *Journal of Quality Technology*, Vol. 12(4) pp 214-219.

E.C. Harrington (1965), "The Desirability Function", *Industrial Quality Control*, Vol 21(10) pp 494-498.

Purpose

To compute a seasonally adjusted series from a non-seasonally adjusted series.

Format

SAMA (*options*) *alist;*
CATNAME = *categories;*
FNAME = *filename;*
METHOD = *method;*
MODE = *mode;*
NAR = *nar;*
NDIFF = *ndiff;*
NMA = *nma;*
NSAR = *nsar;*
NSDIFF = *nsdiff;*
NSMA = *nsma;*
PERIODS = *periods;*
TITLE = *title;*
VLIST = *vlist;*

Input

options	optional, print options.
alist	literal, required, adjusted series list.
categories	literal, optional, save extension list.
filename	literal, optional, specification file name.
method	literal, optional, normalization method (GEOM).
mode	literal, optional, transformation mode (NONE).
nar	numeric, optional, number of autoregressive terms (0).
ndiff	numeric, optional, degree of differencing (0).
nma	numeric, optional, number of moving average terms (0).
nsar	numeric, optional, number of seasonal AR terms (0).
nsdiff	numeric, optional, degree of seasonal differencing (0).
nsma	numeric, optional, number of seasonal MA terms (0).
period	numeric, optional, periodicity.
title	literal, optional, title.
vlist	literal, required, variable list of original series.

Values in parentheses are the default values.

Remarks

SAMA computes a seasonally adjusted series by a moving average method. The names of the unadjusted series are given in *vlist* for each series, a seasonally adjusted series is estimated and stored in the corresponding name in *alist*. These vectors can then be used as if they had been created with a GENR statement. There must be sufficient workspace for the entire series to be stored in core. Missing values are not permitted.

The default periodicity is determined by the type of GAUSSX workspace originally specified in the CREATE statement. Thus if (q) were specified, the default periodicity would be 4 since the workspace is set up for quarterly data; for monthly data (m), the default periodicity is 12. For annual and undated data, the periodicity must be explicitly specified.

GAUSSX uses three seasonal smoothing methods:

ARITH	Moving average method with the seasonal factors normalized arithmetically.
[GEOM]	Moving average method with the seasonal factors normalized geometrically.
X12	ARIMA method using the Census X12 ARIMA program.

Moving Average Method The ratio of each observation in the series to the moving average around that observation is averaged to derived each element of the seasonal factors - there are *period* elements in the seasonal factors. These factors are then normalized depending on the choice of METHOD. The seasonal adjusted series are computed by dividing the original series by the seasonal factors.

ARIMA Method X-12 is a seasonal adjustment program developed by the US Census Bureau for quarterly and monthly data. It is based on the X-11 seasonal adjustment method, which is widely used by statistical agencies throughout the world. X12 also provides the ability to forecast seasonally adjusted time series. GAUSSX provides support for X12 from within the SAMA procedure, and uses the code made available by the Census Bureau. Temporary files are written to the TEMP folder.

The GAUSSX implementation of X12_ARIMA allows for three modes of operation:

- Allow X12 to automatically determine the ARIMA parameters. No specification file is required, and the seasonally adjusted series are returned in *alist*. See Example 3.
- The ARIMA parameters are specified by the user. Thus the order of the AR (*nar, nsar*) and MA (*nma, nsma*) components are required, as well as the degree of differencing (*ndiff, nsdiff*). In addition an optional series transformation is permitted using the MODE option. Valid values for *mode* are NONE, LOG, INVERSE, SQRT, LOGISTIC. No specification file is required, and the seasonally adjusted series are returned in *alist*. See Example 4.
- All X12 input is given in *filename*, a user defined specification file. The non-seasonally adjusted series given in *vlist* will be written to the file sama_x12.asc on the TEMP folder. The user defined specification file will generate output series with the SAVE command; these extensions must be specified in *categories*, and the series will be saved in *alist*. See Example 5.

Print options include b — brief output, p — pause after each screen display, and q — quiet - no screen or printed output.

Examples of the use of X12 are given in test37.prg. The file gauss\gsx\doc\x12_-doc.htm provides a link to documentation for Census X12.

Example

```
1.   CREATE (Q) 19741 19814;
       .
       .
     SAMA gnpqa invqa;
        VLIST = gnpq invq;

2.   SAMA consuma;
        VLIST = consum;
        METHOD = arith;
        PERIODS = 5;

3.   SAMA (p,b) equipa;
        VLIST = equip;
        METHOD = X12;
```

```
4.   SAMA (q) equips;
        VLIST = equip;
        METHOD = X12;
        MODE = log;
        NAR = 2;  NDIFF = 1;  NMA = 1;
        NSAR = 2; NSDIFF = 1; NSMA = 2;

5.   SAMA salesa factors;
        VLIST = sales;
        FNAME = g:\\gauss\gsx\x12\s11.spc;
        METHOD = X12;
        CATNAME = d11 d10;
```

In the first example, two seasonally adjusted series are created – gnpqa is created from gnpq and invqa from invq. Since PERIODS is not specified, the periodicity is taken as 4, based on the type of workspace (quarterly data), with geometric scaling as the default.

In the second example, consuma is the seasonally adjusted series created from consum; arithmetic scaling is specified, and a periodicity of 5.

Examples 3 through 5 demonstrate X12 smoothing. In example 3, the seasonally adjusted series equipa is created from equip, using parameters selected by X12. The b option generates brief output. Example 4 is similar, except the user defines the ARIMA model to be used, as well as a data transformation (MODE = log). The q option generates no (quiet) output. In example 5, the user defines a specification file (.spc) which provides the required information to X12 - thus the full power of the Census X12 is available. Series d11 and d10 are to be saved; they are specified in the CATNAME option, and are saved under salesa and factors respectively.

See Also ARIMA, CREATE, DIVISIA, PRIN

Purpose

To save the current GAUSSX work file onto disk.

Format

SAVE *vlist;*
FNAME = *filename;*
FMTLIST = *fmtopts;*
OPLIST = *progopts ;*

Input

vlist	literal, optional, variable list.
filename	literal, required, the name of an external file.
fmtopts	literal, optional, format options.
progopts	literal, optional, program options.

Remarks

The SAVE statement saves a GAUSSX data set onto disk, in the specified file, using the path specified in the data file option in the GAUSSX menu. An extension for the file name is not required. The data set consists of all the vectors named in *vlist*, plus the GAUSSX vectors C, _ID, and _SAMPLE. A GAUSS data set consisting solely of the vectors given in *vlist* can be specified by using the option OPLIST = GAUSS. If *vlist* is not specified, all the vectors currently defined in the current GAUSSX workfile will be saved.

The data set contains all the current vectors, for the observations specified in the current sample. To save the entire workfile, first specify a SMPL range equivalent to the range specified in the CREATE command. Vectors with missing values are saved as is.

If *filename* includes an .xls extension, the file will be saved as an Excel spreadsheet, with the variable names as headers in the first row. For other extensions, the file will be saved in an ASCII format on the data file subdirectory. The FMTLIST option can be used to specify the format for ASCII files.

Example

```
1.   SAVE; FNAME = sfile;

2.   SAVE x1 x2 x3; FNAME = sfile;

3.   SAVE x1 x2; FNAME = sfile.asc; FMTLIST = width=12;

4.   SAVE x1 x2; FNAME = xfile.xls;
```

In example 1, a GAUSSX data set is created containing all the current vectors, and saved in a file called `sfile`.

In the second example, only vectors `x1`, `x2` and `x3` are saved.

In example 3, an ASCII file called `sfile.asc` containing `x1` and `x2` is created with a field with of 12.

An Excel file with the same variables is created in example 4.

See Also CREATE, DROP, FMTLIST, KEEP, OPEN

Purpose To save the current symbolic gradient and Hessian procedures.

Format
```
SAVEPROC ;
   GRADIENT = &gradname;
   HESSIAN = &hessname;
```

Input
&*gradname* string, optional, procedure name.
&*hessname* string, optional, procedure name.

Remarks The SAVEPROC statement saves the gradient and Hessian procedures created using &symgrad and &symhess. These procedures can thus be stored and reused without having to recreate them each time.

The procedures are saved as ASCII files with a prc extension on the GAUSSX data path.

Example
```
ML (p,i,s) eq1 eq2;
   METHOD = nr bhhh nr;
   GRADIENT = &symgrad;
SAVEPROC ;
   GRADIENT = garchp;
```

This example stores the symbolic gradient as a GAUSS procedure called garchp.prc on the GAUSSX data path.

See Also LOADPROC, ML

Purpose Creates a vector of log likelihoods for a smallest extreme value process.

Format `z = SEV ( y, indx, pvec );`

Input

y	literal, dependent variable.
indx	literal, location index.
pvec	literal, scale parameter.

Output

z	Vector of log likelihoods.

Remarks The expected value of loc_i is parameterized as:

$$E(loc_i) = indx_i.$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β and *pvec*, are estimated using maximum likelihood; thus this can be used for linear or non-linear models. The scale parameter must be positive. The expected value of location is the mode of *y*.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the “General Notes for Non-Linear Models” under NLS. An example is given in test57.prg.

Example

```
PARAM b0 b1 b2;
PARAM scale; value = 1;
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;

1  FRML eq1 llfn =  sev(fail,indx,scale);
   ML (p,i) eq0 eq1;
     METHOD = nr bhhh bhhh;

2  FRML eq2 llfn =  sev(fail~censor,indx,scale);
   ML (p,i) eq0 eq2;
```

In example 1, a smallest extreme value model is estimated using maximum likelihood, with the index defined in eq0, and the log likelihood in eq1. Example 2 shows the same estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose Control over the simulated annealing process.

Format
GAUSSX COMMAND *vlist;*
METHOD = *methodlist;*
SIMANN = *controllist;*

Input

vlist	literal,	required, variable list.
methodlist	literal,	required, algorithm list.
controllist	literal,	optional, list of control options.

Remarks Simulated annealing is a search algorithm used during nonlinear estimation that permits both uphill and downhill movements during the optimization process. It can be very useful for testing if one is at a global optimum, as well as for situations when one gets a "failure to improve likelihood" error message. It is also very robust when parameter upper or lower bounds are encountered.

Simulated annealing can be implemented as a step method during the estimation of nonlinear systems - FIML, GMM, ML, and NLS.Thus one can use SA for the first two elements of *methodlist* to find the parameter values, and then use one of the other stepsize algorithms for the final method to evaluate the Hessian. However, it is considerably slower than the other stepsize methods, although the speed can be adjusted by adjusting the control options. SA can be used with constrained optimization - in this case a penalty function is used to constrain the parameters to the feasible region.

Control over the SA options is provided by the SIMANN option; this consists of a 4 element vector *controllist*; these elements are:

1. The initial temperature. The higher the temperature, the larger the step length. Steps that increase the objective function are always accepted; steps that decrease it are accepted on the basis of the Metropolis criteria. Default = 5.
2. Temperature reduction factor, applied at each new iteration. Default = .85.
3. Number of step length adjustments per iteration. The step length is adjusted so that approximately half of all evaluations are accepted. Default = 100.

4. Number of cycles. During this number of cycles, the number of acceptances is recorded as an input into the step length adjustment. Default = 20.

Example

```
NLS(p,i) eq1;
   METHOD = gauss sa nr ;
   SIMANN = 8 .75 20 10;
   MAXIT = 40;
```

This example would undertake non-linear least squares on eq1 using gauss as the initial step method, sa the remaining steps, except for the final step (where one needs the Hessian) which is estimated using Newton-Raphson (nr). The SA process uses an initial temperature of 8, a reduction factor of .75, 20 step length adjustments, and 10 cycles.

See Also FIML, GMM, ML, NLS,

Purpose To specify which observations are to be included in subsequent operations.

Format SMPL *vlist ;*

Input *vlist* required, variable name or observation list.

Remarks There are two types of SMPL statement. The first type, shown in examples 1 to 3 below, specifies pairs of first and last observations to be included in the current sample. In the second type, *vlist* is the name of a vector created in a previous operation; example 4 depicts such a case. Observations are included for those cases for which `svector` takes a value of unity - all observations for which `svector` takes any other value (including missing values) are excluded.

The range specified must fall within the range specified in the CREATE statement. Similarly, the range arguments must be of the same type (annual, quarterly, etc) as the arguments used in the CREATE statement.

Example

```
1.   SMPL 1973 1980;

2.   SMPL 197201 197212 197401 197412;

3.   n1 = 1971;  n2 = 1984;
     SMPL n1 n2;

4.   SMPL svector;
```

In example 1, eight observations are included in the current sample – 1973 to 1980. In the second example, two years of monthly observations are specified, for 1972 and 1974. Variables can be used as arguments, as is shown in example 3. Example 4 depicts the second type of SMPL statement, where `svector` was previously defined either by a GENR or LOAD statement.

See Also CREATE

Purpose

Computes the solution of a non-linear equation system.

Format

```
SOLVE vlist ;
   ENDOG = endlist;
   EQNS  = elist;
   EQSUB = macrolist;
   MAXIT  = maxit;
   METHOD = methname;
   MODE   = mode;
   JACOB  = Jacobian;
   TOL    = tolerance;
```

Input

vlist	literal, required, variable list.
endlist	literal, required, endogenous variable.
elist	literal, required, equation list.
macrolist	literal, optional, macro equation list.
maxit	numeric, optional, maximum number of iterations (20).
methname	literal, optional, stepsize method (DIFFER).
mode	literal, optional type of simulation (STATIC).
Jacobian	literal, optional, Jacobian.
tolerance	numeric, optional, param. convergence tolerance (.001).

Values in parentheses are the default values.

Remarks

The SOLVE command solves for the endogenous variables of a system of equations. The order of the fitted values corresponds to the order of the variable in the ENDOG list. Note that the ENDOG and EQNS commands are required. The forecast variables are available for use in the same way as a variable created through FORCST.

The type of simulation can be set by the MODE options. Simulation occurs in two modes:

[STATIC]	Lagged dependent variables take their historical values.
DYNAMIC	Lagged dependent variables take their simulated values.

Two step size methods are available using the METHOD option:

[DIFFER]	Finite difference of Jacobian.
BROYDEN	Broyden secant approximation.

The initial starting values used are the historical values of the endogenous variables. Thus for future periods, a best guess should be used for each of the endogenous variables. GAUSSX will evaluate the Jacobian if it is not specified, though this increases the computation time.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test03.prg.

Example

```
FRML eq1 y1 = a0 + a1*x1 + a2*y2;
FRML eq2 y2 = b0 + b1*x3 + b2*y1^2 + b3*lag(y1,1);
SMPL 2  12;

1.  SOLVE  y1s y2s;
       ENDOG = y1 y2;
       EQNS  = eq1 eq2;

2.  SOLVE  y1hat y2hat;
       ENDOG = y1 y2;
       EQNS  = eq1 eq2;
       MODE  = dynamic;
       JACOB =  1       -a2
              -2*b2*y1   1;
```

In the first example, the roots of a system of equations (which have been previously estimated) are determined by SOLVE. Since there is a lagged endogenous variable, the first observation is dropped using the SMPL command. The value of the two endogenous variables are printed for each observation. GAUSSX takes care of the Jacobian. Note that this is a static simulation, the lagged y1 are the historic values.

In the second example, the Jacobian is specified by the user. A dynamic

simulation takes place. The solution to the equation system is stored in the vectors `y1hat` and `y2hat`.

See Also FORCST

Purpose
Creates a power spectrum from a series.

Format
y = SPECTRAL (*x*, *wind*);

Input

x	*N*x1 real input vector.
wind	string, spectral window.

Output

y	Nx1 power spectrum vector.

Remarks
SPECTRAL returns the power spectrum for a series *x*. The power spectrum *y* consists of a vector of half the length of *x*, the remainder being set to missing. A spectral window can be specified - the available windows are BARTLETT, HANNING, PARZEN, UNIFORM and WELCH. The default is UNIFORM. Power spectral density uses the FFT routines, which uses zeros to pad to powers of 2. This works well for smooth windows, but creates a discontinuity for UNIFORM windows - hence, in this case, a slow FFT is used to avoid padding.

SPECTRAL is pure GAUSS code, and can be used independently of GAUSSX.

An example of SPECTRAL demonstrating both the generation of a periodogram and filtering is given in test24.prg.

Example

```
library gaussx ;
y = spectral(x,Parzen)'
```

This creates y, the power spectra of x, with a Parzen spectral window.

Source
GXPROCS.SRC

See Also
WINDOW

References
Press, W.H. *et. al.* (1986), *Numerical Recipes*, Cambridge University Press, New York.

Purpose

Computes measures for the specified distribution.

Format

```
llf  = FN_LLF  ( x, p1, p2, p3 );
pdf  = FN_PDF  ( x, p1, p2, p3 );
cdf  = FN_CDF  ( x, p1, p2, p3 );
cdfi = FN_CDFI ( p, p1, p2, p3 );
y    = FN_RND  ( n, k, p1, p2, p3 );
```

Input

FN	string, the name of the probability distribution - see below.
x	*N*x*K* matrix, the argument to the specified distribution.
p	*N*x*K* matrix of probabilities.
n	scalar, row dimension.
k	scalar, column dimension.
p1	*N*x*K* matrix or scalar, first parameter for the specified distribution.
p2	*N*x*K* matrix or scalar, optional second parameter for the specified distribution.
p3	*N*x*K* matrix or scalar, optional third parameter for the specified distribution.

Output

llf	*N*x*K* matrix of log likelihoods
pdf	*N*x*K* matrix of probabilities
cdf	*N*x*K* matrix of cumulative probabilities
cdfi	*N*x*K* matrix of inverse cumulative probabilities
y	*N*x*K* matrix of random variates

Remarks

Five functions are available for each supported distribution - the log likelihhod, the probability denisty function, the cumulative density function, the inverse cumulative density function and random variates. Each distribution is characterized by up to three parameters - *p1*, *p2* and *p3* - the number of parameters depends on the specified distribution. Ususally these parameters are scalars; otherwise they must be conformable with the argument.

The following distributions are supported in STATLIB:

BETA — The beta distribution takes a continuous argument, *x*, which must lie in the interval [0 1], and two positive shape parameters.

BETA4 The four parameter beta distribution takes a continuous positive argument, *x*, two positive shape parameters, a lower bound and an upper bound.

BOXCOX The BoxCox distribution takes a continuous argument, *x*, and three parameters, *p*1 the location parameter, *p*2 the positive scale parameter, and *p*3 the BoxCox transformation parameter.

BERNOULLI The Bernoulli distribution takes an integer argument, *x*, which is either zero or unity, and a single probability parameter,*p*1, which must lie in the interval [0 1].

BINOM The binomial distribution takes an integer, non-negative argument, *x*, and two parameters,*n*, which is a positive integer, and *p*, which must lie in the interval [0 1]. It returns the probability of *x* successes in *n* independent trials, where *p* is the probability of success in any given trial.

BURR The Burr distribution takes a continuous positive argument, *x*, a positive scale parameter, and two positive shape parameters.

CAUCHY The Cauchy distribution takes an unbounded continuous argument, *x*, and two parameters, *p*1, the median, and *p*2, a positive scale parameter. It has no moments. It is infinitely divisible, since the mean of *n*-independent Cauchy distributions is also Cauchy.

CHISQ The Chi-squared distribution takes a non-negative continuous argument, *x*, and a single positive shape parameter, *v*, the degree of freedom. While *v* is normally taken as integer, STATLIB implements CHISQ with continuous *v*. The sum of squares of *v* observations, each independently distributed standard normal, is distributed chi-squared with *v* degrees of freedom.

CHISQ_SCALED The scaled Chi-squared distribution takes a non-negative

continuous argument, x, a positive scale parameter, and a single positive shape parameter, v, the degree of freedom. While v is normally taken as integer, STATLIB implements CHISQ_SCALED with continuous v.

ERF The ERF distribution takes a continuous unbounded argument, x, and a single positive scale parameter. The ERF distribution is similar to the normal distribution, but with a zero mean.

EXPON The exponential distribution takes a non-negative continuous argument, x, and a single positive scale parameter, λ. The exponential function is used to model waiting times.

F The F distribution takes a continuous non-negative argument, x, and two positive shape parameters, $v1$ and $v2$, the degrees of freedom. While $v1$ and $v2$ are normally taken as integer, STATLIB implements F with continuous shape parameters.

F_SCALED The scaled F distribution takes a continuous non-negative argument, x, a positive scale parmeter, and two positive shape parameters, $v1$ and $v2$, the degrees of freedom. While $v1$ and $v2$ are normally taken as integer, STATLIB implements F_SCALED with continuous shape parameters.

FATIGUELIFE The fatigue life distribution (or Birnbaum Saunders distribution) takes a continuous positive argument, x, and two parameters, $p1$, a positive scale parameter, and $p2$, a positive shape parameter. It is used to model the lifetime of a device suffering from fatigue.

FOLDEDNORMAL The folded normal distribution takes a continuous positive argument, x, and two parameters, $p1$, a positive mean parameter, and $p2$, a positive scale parameter. If y is distributed normally, then $|y|$ is distributed as folded normal.

FRECHET The Frechet distribution takes a continuous positive argument,

x, and two parameters, *p*1, a positive scale parameter, and *p*2, a positive shape parameter. The Frechet distribution is a special case of the generalized extreme value distribution

GAMMA The gamma distribution takes a continuous non-negative argument, *x*, a positive scale parameter, *p*1, and a positive shape parameter, *p*2. The gamma distribution is typically used in reliability models.

GED The generalized error distribution (or exponential power distribution) takes a continuous argument, *x*, and three parameters, *p*1, a location parameter, *p*2, a positive scale parameter, and *p*3, a positive shape parameter. This includes the Laplace distribution (*p*3 = 1) and the normal distribution (*p*3 = 2).

GENGAMMA The generalized gamma distribution takes a continuous non-negative argument, *x*, a positive scale parameter, *p*1, and two positive shape parameters, *p*2 and *p*3. This is a generalization of the gamma distribution, and includes the exponential, log normal, Maxwell and Weibull distributions as special cases.

GENLOGISTIC The generalized logistic distribution takes a continuous argument, *x*, *x*, and three parameters, *p*1, a location parameter, *p*2, a positive scale parameter, and *p*3, a positive skew parameter (< 1 for left skew, > 1 for right skew). It is used to model extremes, such as maximum rainfall.

GENPARETO The generalized Pareto distribution takes a continuous positive argument, *x* (*x* > *p*1), and three parameters, *p*1, a positive location parameter, *p*2, a positive scale parameter, and *p*3, a positive shape parameter.

GEOMETRIC The geometric distribution takes a non-negative integer argument, *x*, and a single probability parameter, *p*, which must lie in the interval [0 1]. It returns the probability of *x* failures before a success, where *p* is the probability of success in any given trial.

GUMBEL The Gumbel (or largest extreme value) distribution takes a continuous unbounded argument, x, a location parameter, $p1$ (the mode), and a positive scale parameter $p2$. The Gumbel distribution is used in the derivation of the MNL model.

HALFNORMAL The half normal distribution takes a continuous positive argument, x, and a positive scale parameter $p1$. It is proportional to the normal distribution, restricted to the positive domain.

HYGEOM The hypergeometric distribution takes a non-negative integer argument, x, and three positive integer parameters, m, k, and n. If there exist k objects of a certain type out of a total of m objects, and n objects are drawn at random without replacement, then pdf is the probability of drawing exactly x items of the specified type.

INVGAUSS The inverse Gaussian distribution takes a continuous positive argument, x, and two parameters, $p1$, the mean, and $p2$, a positive scale parameter.

JOHNSON_SB The Johnson SB distribution takes a continuous bounded argument, x, and four parameters, $p1$, a location parameter, $p2$, a positive scale parameter, $p3$ a shape parameter, and $p4$, a positive shape parameter.

JOHNSON_SL The Johnson SL distribution takes a continuous bounded argument, $x > p1$, and four parameters, $p1$, a location parameter, $p2 = 1$, a scale parameter, $p3$ a shape parameter, and $p4$, a positive shape parameter.

JOHNSON_SU The Johnson SU distribution takes a continuous unbounded argument, x, and four parameters, $p1$, a location parameter, $p2$, a positive scale parameter, $p3$ a shape parameter, and $p4$, a positive shape parameter. x is bounded by $p1$ and $p1 + p2$

LAPLACE The Laplace distribution takes an unbounded continuous argument, x, and two parameters, $p1$, the mean, and $p2$, a positive scale parameter. The Laplace distribution results from the

difference of two independent identically distributed exponential random variables.

LEV Largest extreme value distribution - see the Gumbel distribution.

LEVY The Levy distribution takes a continuous positive argument, x, and a positive scale parameter, $p1$. Some random walks can be modeled with this distribution.

LOGGAMMA The log-gamma distribution takes a continuous non-negative argument, x, a positive scale parameter, $p1$, and a positive shape parameter, $p2$.

LOGARITHMIC The logarithmic distribution is a one parameter generalized power series distribution. It takes a non-negative integer argument, x, and a single probability parameter,$p1$, which must lie in the interval [0 1].

LOGISTIC The logistic distribution takes a continuous argument, x, and two parameters, $p1$, the mean, and $p2$, a positive scale parameter. It has longer tails than the normal distribution.

LOGLOG The log logistic distribution takes a continuous positive argument, x, and two parameters, $p1$ and $p2$, the mean and scale of the associated logistic distribution. $p2$ must be positive.

LOGNORM The log-normal distribution takes a continuous positive argument, x, and two parameters, μ and σ, the mean and standard deviation of the associated normal distribution. σ must be positive. If y is log-normal, then $\ln(y)$ is normal. It is used for variates which can only take positive values, such as the size of particles in an emulsion.

MAXWELL The Maxwell Boltzmann distribution takes a continuous positive argument, x, and a single positive scale parameter, $p1$.

NEGBIN The negative binomial distribution takes an integer, non-negative argument, x, and two parameters, s, which is a non-negative

integer, and p, which must lie in the interval [0 1]. pdf is the probability of x failures before the sth success, where p is the probability of success in any given trial. STATLIB implements NEGBIN with continuous s.

NCCHISQ The non-central Chi-squared distribution takes a non-negative continuous argument, x, a positive shape parameter, v, the degree of freedom, and a positive non-centrality parameter, λ. While v is normally taken as integer, STATLIB implements CHISQ3 with continuous v.

NCF The non-central F distribution takes a continuous non-negative argument, x, two positive shape parameters, $v1$ and $v2$, the degrees of freedom, and a positive non-centrality parameter, λ. While $v1$ and $v2$ are normally taken as integer, STATLIB implements F with continuous shape parameters.

NCT The non-central T distribution takes an unbounded continuous argument, x, a positive shape parameter, v, the degree of freedom, and a positive non-centrality parameter, λ. While v is normally taken as integer, STATLIB implements CHISQ3 with continuous v.

NORMAL The normal distribution takes a continuous unbounded argument, x, and two parameters, $p1$, the mean, and a positive scale parameter, $p2$, (the standard deviation).

PARETO The Pareto distribution takes a continuous positive argument, x ($x > p1$), and two parameters, $p1$, a positive location parameter, and $p2$, a positive shape parameter. It is used to model income distribution.

PEARSON The Pearson type III distribution takes a continuous nonnegative argument, x, and three parameters, $p1$, the location, $p2$, a positive scale parameter, and $p3$, a positive shape parameter.. This distribution is very general, and includes as special cases the beta, gamma, normal and t distributions. This family

is modeled assuming zero mean. The standard normal distribution corresponds to $p1 = 1$, $p2 = 1$, and $p3 = 0$.

PERT The PERT distribution takes a continuous argument, x, which must lie in the interval [a c], and three parameters, a, the minimum, b, the mode, and c, the maximum, $c > b > a$.

POISSON The Poisson distribution takes a non-negative integer argument, x, and a single positive parameter, λ, the mean. The pdf is the probability of x events occurring within a period, where λ is the expected number of events in that period.

POWER The power distribution takes a continuous argument, x, which must lie in the interval [a b], and three parameters, a, the minimum, b, the maximum, and a positive shape parameter, v.

RAYLEIGH The Rayleigh distribution takes a continuous argument, x, and a single positive scale parameter, $p1$. The Rayleigh distribution is equivalent to a Weibull distribution with shape = 2.

RECIROCAL The reciprocal distribution takes a continuous argument, x, which must lie in the interval [a b], and two parameters, $a > o$, the minimum, and b, the maximum.

RECTANGULAR The rectangular distribution takes an integer argument, x, which must lie in the interval [a b], and two parameters, a, the minimum, b, the maximum, where $b > a$. The pdf has the same probability at each point in the specified interval.

SEV The smallest extreme value distribution takes a continuous unbounded argument, x, a location parameter, $p1$ (the mode), and a positive scale parameter $p2$.

SKEWNORMAL The skew normal distribution takes a continuous argument, x , and three parameters, $p1$, a location parameter, $p2$, a positive scale parameter and $p3$, a skew parameter (-ve for left skew, +ve for right skew).

STEP The step distribution takes an integer argument, x, which must lie in the interval [a b], and three parameters, a, the minimum, b, the maximum, and s, the step,

STUDENTS_T The Student's t distribution takes an unbounded continuous argument, x, and a single positive shape parameter, v, the degree of freedom. While v is normally taken as integer, STATLIB implements Student's t with continuous v. The Student's t distribution tends to the normal distribution as $v \rightarrow \infty$.

T_SCALED The scaled T distribution takes an unbounded continuous argument, x, a location parameter, μ, a positive scale prarmeter, α, and a positive shape parameter, v, the degree of freedom. While v is normally taken as integer, STATLIB implements T_SCALED with continuous v. The T_SCALED distribution is a generalization of the Student's t distribution.

TRIANGULAR The triangular distribution takes a continuous argument, x, which must lie in the interval [a b], and three parameters, a, the minimum, b, the maximum, and c, the mode. $b > c > a$.

UNIFORM The uniform distribution takes a continuous argument, x, which must lie in the interval [a b], and two parameters, a, the minimum, b, the maximum, where $b > a$. pdf has the same probability at each point in the specified interval.

VONMISES The Von Mises distribution takes a continuous non-negative argument, x, which must lie in the interval [0 2π], and two positive parameters, $p1$, the location (also bounded as x), and $p2$, the scale.

WEIBULL The Weibull distribution takes a continuous non-negative argument, x, and two positive parameters, $p1$, the scale, and $p2$, the shape. The type 1 extreme value distribution is derived from the Weibull distribution.

STATLIB is pure GAUSS code, and can be used independently of GAUSSX.

The PDF and CDF for each distribution is given in Appendix A.8. Random variates use the KissMonster pseudo random number algorithm, hence use _KMseed to set the seed.

The parameters of a statlib distribution can be estimated using ML. Note that for those distributions that are defined for a positive argument (eg. Weibull), a threshold parameter can be estimated.

An example ofSTATLIB is given in test64.prg.

Example

```
library gaussx ;
x = seqa(0,.2,6);
a = 2; b = 4;
p = beta_pdf(x,a,b);

 x' =  0.0000  0.2000  0.4000  0.6000  0.8000  1.0000

 p' =  0.0000  2.0480  1.7280  0.7680  0.1280  0.0000
```

This computes the probability given the argument x and parameters a and b for the beta pdf.

Source STATLIB.SRC

See Also CDF, CDFI, PDF, QDFN, RND

References

Evans, M., N. Hastings and B. Peacock (1993), *Statistical Distributions*, 2nd ed. John Wiley, New York.

Purpose This procedure undertakes stepwise regression.

Format { *xmat, namestr* } = STEPWISE (*dta, dtaname, oplist*);

Input

dta	*N*x*K* matrix of data.
dtaname	*K*x1 string array of names, or 0.
oplist	4x1 vector of program options.

Output

xmat	matrix of selected data
namestr	name of each column of *xmat*

Remarks Stepwise linear regression examines variables incorporated in the model at every stage of the regression. A variable which may have been the best choice to enter the model at an early stage may later become non-significant because of the relationships between it and other variables now in the regression. Once a variable is proven to be non-significant, it is removed from the model. This process continues until no more variables can be accepted and no more can be rejected.

dta is the *N*x*K* data matrix. The first column is taken as the dependent variable, and the remaining columns are the independent variables. A constant is not necessary. The names of each variable is provided in the *K*x1 string array *dtaname*; default values are used if *dtaname* equals zero.

The program control options are specified in the 4 element vector *oplist*. The options available are:

1	*pfin* Probability of F_to_Enter
2	*pfout* Probability of F_to_Remove
3	Scaling: 0 - none; 1 - standardized (zero mean and unit variance); 2 - ranged (-1 to +1).
4	Hierarchy: 0 - linear only; 1 - liner and quad; 2 - linear and cross; 3 - linear and cross and quad.

The process of determining whether or not a variable is significant is based on the F-statistic; the user provides the statistical significance level (alpha) for variables entering and exiting the model. A value of alpha near 1.0 for *pfin*

allows variables to easily enter the model, while a value of alpha near 0.0 for *pfin* prevents variables from entering the model. Similarly, a value of alpha near 1.0 for *pfout* prevents variables from easily leaving the model, while a value of alpha near 0.0 for *pfout* enables variables to easily be removed from the model. Alpha values of (0.05,0.05) are recommended. Alpha values of (0.999,0.999) closely approximate ordinary least squares. Note that for consistency, the probability of exit must be less than the probability of entry.

The data is initially scaled based on the third element of *oplist*. The fourth element (hierarchy) determines the type of expansion. With hierarchy set to zero, STEPWISE uses a constant and the data provided in *dtaname*. Otherwise, the data can be expanded to include cross and/or quad terms by setting hierarchy to the requisite value.

Although stepwise methods can find meaningful patterns in data, it is also notorious for finding false patterns.

STEPWISE is pure GAUSS code, and can be used independently of GAUSSX.

An example of stepwise is given in test63.prg.

Example

```
library gaussx ;
oplist = { .4 .25 0 1 };
{xnew, xnames} = stepwise(y~xmat, 0, oplist);
```

This example shows how a stepwise regression is applied to a matrix of potential explanatory variables `xmat`, expanded to include quad terms. Values of .4 and .25 are used for the F statistic probability of entry and exit, respectively.

Source STEPWISE.SRC

See Also XPAND

References Miller, A.J. (1966), “The convergence of Efroymson’s stepwise regression algorithm”, *American Statistician*, Vol. 50(2), pp. 180-181.

Purpose To store global variables in a GAUSSX workfile.

Format STORE *varlist;*
VLIST = *matname;*

Input

varlist	literal, required, variable list.
matname	literal, optional, matrix name.

Remarks The STORE command instructs GAUSSX to access the named global vectors, and store them on the current GAUSSX workfile. The current SMPL statement remains in effect. This command allows vectors created using GAUSS commands to be accessed by GAUSSX . The VLIST option allows the user to access the global matrix *matname*, and store each column as a GAUSSX variable.

STORE clears the named series - they can now only be accessed using GAUSSX commands. To make a series in the GAUSSX workspace into a global variable, use the FETCH command.

See the example in test06.prg, and also the discussion in Appendix C.

Example

```
1.   Test of Henon equation;
     num = 300;    = -1.4;  b = .3;
     x = zeros(num,1);  y = zeros(num,1);
     i = 2;   do until i > num;
        j = i-1;
        x[i,1] = 1 + y[j,1] + a*x[j,1]^2;
        y[i,1] = b*x[j,1];
        i = i+1;
     endo;
     STORE x y;
     SMPL 100 300;
     OLS x c y;

2.   z = rndu(100,3);
     STORE z1 z2 z3;
        VLIST = z;
```

The first example shows how global vectors created by ordinary GAUSS statements can be incorporated into a GAUSSX workspace. In this case, **x** and **y** are the state vectors of a Henon map, and once they have been stored they are subsequently treated as ordinary GAUSSX vectors.

The second example shows how the columns of a GAUSS matrix **z** can be stored as GAUSSX variables.

See Also FETCH, GAUSS

Purpose

Estimates the coefficients of a system of a seemingly unrelated equations

Format

SURE (*options*) *elist;*
PDL = *pdllist;*
METHOD = *methname;*
TITLE = *title;*
WINDOW = *windowtype;*

Input

options	optional, print options.
elist	literal, required, equation list.
pdllist	literal, optional, options for PDL.
methname	literal, optional, covariance method (NONE).
title	string, optional, title.
wtname	literal, optional, weighting variable.
windowtype	literal/numeric, optional, spectral window.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
LLF	Log likelihood.
VCOV	Parameter Covariance matrix.
COVU	Residual Covariance matrix.

Remarks

The SURE command estimates a system of linear equations (a stacked equation system) in two stages. In the first stage each equation is estimated using OLS; then using the estimated variance covariance matrix of residuals, the system is estimated using generalized least squares. Non-linear SURE can be carried out using the NLS command, and specifying MAXITW = 1;

See the "General Notes for Linear Model" under OLS. An example is given in test02.prg.

Example

```
1.   SURE eq1 eq2 ;

2.   SURE (d,p,s,v) eq1 eq2 eq3;
```

The first example estimates the system of linear equations specified in previous FRML commands.

The second example estimates the three equation system; execution pauses (p) after each screen display, descriptive (d) and diagnostic (s) statistics are provided, and the variance-covariance (v) matrix is displayed.

See Also

FRML, OLS, PDL, TITLE, WEIGHT, WINDOW

References

Zellner, A. (1962), “An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests of Aggregation Bias”, *JASA*, Vol. 57, pp. 348-368.

Purpose Computes non-parametric estimates of survival and hazard rates.

Format
SURVIVAL (*options*) *varlist ;*
CENSOR = *cenvar;*
GROUP = *grouplist;*
METHOD = *method;*
MODE = *measure;*
RANGE = *rangelist;*
TITLE = *title;*
VLIST = *depvar;*

Input

options	optional, print options.
varlist	literal, required, variable list.
cenvar	literal, optional, censor variable.
grouplist	literal, optional, group variable list.
method	literal, optional, algorithm
measure	literal, optional, survival measure.
rangelist	literal, optional,range list.
title	string, optional, title.
depvar	literal, required, survival time.

Output
_MTTF Distribution characteristics.

Remarks The SURVIVAL command computes non-parametric estimates of survival or hazard rates, and produce a tabular output of the rate, the standard error and confidence bands. Duration or survival models typically model the duration of an event, or the time to failure.

In the default, there is no censoring, and only *depvar* needs to be specified. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *cenvar* is specified, with each element taking a value of unity if the unit was censored, else zero.

varlist consists of up to four elements - the statistic, the standard error, and the lower and upper confidence bands. The survival measure bands are

lower truncated at zero. If *varlist* consists of less than four elements, then only these elements will be evaluated.

The non-parametric algorithm is set in *method*. The available algorithms are:

KAPLAN	Kaplan-Meier. This is the default.
NELSON	Nelson-Aalen.

The survival measure is set in *measure*. The available measures are:

SURVIVAL	The survival rate. This is the default.
CUMFAIL	The cumulative failure rate.
CUMHAZARD	The cumulative hazard rate.
HAZARD	The hazard rate.

Stratified data can be sequentially estimated by using *grouplist*; however this is descriptive only, and *varlist* is ignored.

Print options include b — print brief output only, d — print descriptive statistics, p — pause after each screen display, and q — no screen display (quiet). A subset of the output can be specified using the RANGE command.

An example of SURVIVAL is given in test58.prg.

Example

```
SURVIVAL (p) ch cherr;
  TITLE = "Censored Cumulative Hazard";
  MODE = cumhazard;
  METHOD = nelson;
  RANGE = 500 2000 ;
  VLIST = failuret;
  CENSOR = cenvar;
PRINT (p) failuret ch cherr;
```

This example generates the cumulative hazard rate (ch) and standard errors (cherr) for the variable failuret with the indicator of right censoring being given in cenvar. The Nelson-Aalen algorithm is used, and output is reported for values of failuret that fall in the specified range.

See Also DURATION, GROUP, TITLE

Purpose Creates a vector of log likelihoods for a stochastic volatility process.

Format `z = SV ( resid, gvec );`

Input

resid	literal, vector of residuals.
gvec	literal, vector of parameters for the SV process.

Output

z	Vector of log likelihoods.
_ht	Vector of the log conditional variance.

Remarks The coefficients of the SV process are estimated using quasi maximum likelihood, based on the Kalman filter algorithm. The SV model is given by:

$$\begin{aligned} y_t &= \sqrt{(h_t)}\epsilon_t \\ \epsilon_t &\sim N(0,1) \\ \log h_t &= \gamma_0 + \gamma_1 \log h_{t-1} + \sigma_v \upsilon_t \\ \upsilon_t &\sim N(0,1) \end{aligned}$$

The first equation describes the structure of the model. Typically, y_t would be the residuals from a previously estimated model. The second and fourth equations specify the distribution of the residuals, while the third equation specifies the structural form of the conditional variance h_t. *gvec* consists of the three parameters in this equation: γ_0, γ_1, and σ_v^2.

The first equation can be transformed into:

$$\log y_t^2 = -1.27 + \log h_t + \eta_t$$

where $E(\eta_t) = 0$ and $V(\eta_t) = .5\pi^2$. This is the measurement equation, while the $\log h_t$ equation is the transition equation. When η_t is approximated by a normal distribution, we have a standard dynamic linear model, that can be estimated using the Kalman Filter algorithm.

resid should be detrended and have zero mean. It is usually not a good idea to estimate structural parameters concurrently with the SV process, since there are significant identification issues.

Note that $\sigma_v^2 > 0$ and stationarity requires that $|\gamma_1| < 1$. Consequently, constrained ML is usually required. An example of SV is given in test30.prg.

See the "General Notes for Non-Linear Models" under NLS.

Example

```
OLS y c x1 x2;
FORCST res;
   MODE = resid;
PARAM gam0 gam1 varnu ;
   VALUE = 3 .6 1 ;
FRML ec1 varnu >= .0001;
FRML ec2 abs(gam1) <= .9999;
FRML eq1 lf = sv(res,gam0|gam1|varnu);
ML(p,i) eq1 ;
   EQCON = ec1  ec2;
hfit = exp(_ht);
STORE hfit;
```

In this example, a stochastic volatility model is estimated using the residuals from an OLS regression. After the constrained ML estimation, the conditional variance `hfit` is derived and stored.

Source KALMANX.SRC

See Also NLS

References

Harvey, A.C., E. Ruiz, and N. Shephard. (1994), "Multivariate Stochastic Variance Models", *Review Econ. Studies*, Vol 61, pp 247-264.

Mills, T. (1999), *The Econometric Modelling of Financial Time Series*, 2nd ed. Cambridge University Press.

Purpose

Undertakes singular value decomposition analysis.

Format

SVD (*options*) *vlist ;*
TITLE = *title;*
VLIST = *elist;*
WEIGHT = *wtname;*

Input

options	optional, print options.
vlist	literal, required, variable list.
title	string, optional, title.
elist	literal, optional, variable list.
wtname	literal, optional, weighting variable.

Output

_SVD	Condition index.

Remarks

Singular value decomposition analysis (SVD) is carried out on the entire matrix of variables specified in *vlist*. Each vector is scaled by the program such that its norm is unity. Lagged variables can be used by specifying the lag in parenthesis. Variables that are specified as logs should first be e-scaled – this is carried out if the vector is included in *elist*. Weighting is available using the WEIGHT option.

Print options includes p — pause after each screen display, and q — quiet - no output displayed.

Example

```
GENR lnx1 = ln(x1);
SVD (p) lnx1 x2 x3;
     VLIST = lnx1;
```

In this example, SVD is carried out on the matrix consisting of the vectors lnx1, x2, and x3. lnx1 is first e-scaled since it is a variable that is measured as a log, and not as a level.

See Also

COVA, TABULATE, TITLE, WEIGHT

References

Belsley, D., E. Kuh, and R. Welsch (1980), *Regression Diagnostics*, Wiley, New York.

Purpose

Constructs a hierarchical table of descriptive statistics.

Format

TABULATE (*options*) *varlist ;*
CATNAME = *categories;*
FMTLIST = *fmtopts;*
GROUP = *grouplist;*
MODE = *statmode;*
TITLE = *title;*
VLIST = *classlist;*
WEIGHT = *wtname;*

Input

options	optional, print options.
varlist	literal, required, variable list.
categories	literal, optional, a list of category names.
fmtopts	literal, optional, format options.
grouplist	literal, optional, group variable list.
statmode	literal, optional, statistic mode list (NUM).
title	string, optional, title.
classlist	literal, required, class variable list.
wtname	literal, optional, weighting variable.

Output

STATS Tabular output.

Remarks

This procedure replicates Proc Tabulate in SAS. It provides a fast and easy way of tabulating a set of data across two class variables according to a specified set of statistics.

TABULATE produces a table for each of the analysis variables specified in *varlist*. The classification levels are derived from each of the two integer variables specified in *classlist*. These two variables specify the row category levels and the column category levels respectively. The labels for each of these class variables is optionally specified in *categories*. If only one variable is specified in *classlist*, a one-way classification is carried out.

For each row and column category, the statistics that are produced are specified in *statmode* the default is NUM. The available statistics are:

NUM	The count of the number of elements in the cell.
SUM	The sum of the analysis variable for the cell.
MIN	The minimum value of the analysis variable for the cell.
MAX	The maximum value of the analysis variable for the cell.
ROW%	The row percentage of the number of elements in the cell.
COL%	The column percentage of the number of elements in the cell.
TOT%	The total percentage of the number of elements in the cell.
MEAN	The mean of the analysis variable for the cell.
STDV	The standard deviation of the analysis variable for the cell.
VAR	The variance of the analysis variable for the cell.
FIT	The expected cell count.
RESID	The raw residual for each cell.
STDRES	The standardized residual for each cell.
ADJRES	The adjusted residual for each cell.
CHISQ	The Chi-squared contribution for each cell.

Print options include p — pause after each screen display, and s — print contingency table statistics. A total is automatically generated.

User defined formatting is available using the FMTLIST option. While a table with a width greater than 80 columns will wrap, the output is set for no wrap, so that the output can be subsequently correctly viewed and/or printed. The numeric values of the table is returned as a global variable called stats.

Weighting is available using the WEIGHT option. Weighting only applies to the analysis variable, and not to counts or percentages.

An example of TABULATE is given in test08.prg.

Example

```
TABULATE (p,s) salary;
   VLIST = agegrp sex;
   CATNAME = age1 age2 age3 age4  male female;
   MODE = num mean min max sum;
   FMTLIST = width= 5 prcn = 0;
```

This generates a table of salary data, with 4 rows, corresponding to four age groups, by two columns, corresponding to two gender groups. For each gender/age category, five statistics are reported - the count (num), and the mean, min, max and sum of salaries, with a user specified format. Contingency table statistics (s) for the age/sex counts are also displayed.

See Also COVA, CROSSTAB, FMTLIST, FREQ, GROUP, TITLE, WEIGHT

Purpose

To compute diagnostic parametric tests.

Format

TEST (*options*) *vlist ;*
BOUND = *level ;*
CENSOR = *cenname;*
ENDOG = *endlist;*
INST = *instlist;*
METHOD = *methname;*
MODE = *modetype;*
OPLIST = *oplist;*
ORDER = *order;*
PERIODS = *periods;*
VALUE = *value;*
VLIST = *rlist;*
WEIGHT = *wtname;*

Input

options	optional, print options.
vlist	literal, required, variable or equation list.
level	numeric, optional, percentage confidence level. (.95)
cenname	literal, optional, censor variable name.
endlist	literal, optional, endogenous variable list.
instlist	literal, optional, list of instruments.
methname	literal, required, diagnostic method.
modetype	literal, optional, algorithm.
oplist	literal, optional, program options.
order	literal, optional, degrees of freedom or maximum lags
periods	literal, optional, subsample range.
value	literal, optional, restriction values or matrix.
rlist	literal, optional, restriction matrix.
wtname	literal, optional, weighting variable.

Remarks

The TEST command provides a number of parametric diagnostic tests; the test chosen is given by *methname*. Parametric tests are based on specific assumptions about the distribution of the population sampled. These tests are generally more powerful than non-parametric tests, which do not make such distributional assumptions.

Print options include p — pause after each screen display, and g — display graph (when available). On-line help (with additional information) is available for each of these tests. The following tests are currently supported:

AD	Anderson-Darling normality test.
ANOVA	Analysis of variance test.
ARCH	Engle's Arch test.
BARTLETT	Bartlett test for homoscedasticity.
BKW	Belsley, Kuh and Walsh SVD test
BP	Breusch Pagan test for homoscedasticity.
CHISQ	χ^2 test.
CHOW	Chow stability test.
DF	Dickey-Fuller unit root test.
EG	Engle-Granger cointegration test.
F	F test.
FTEST	Linear restriction test.
GRANGER	Granger causality test.
HANSEN	Hansen test of overidentifying restrictions.
HAUSMAN	Hausman specification test.
JB	Jarque-Bera normality test.
JOHANSEN	Johansen cointegration test.
JTEST	Davidson and MacKinnon J-test for restrictions.
KPSS	Kpss stationarity test.
LBQ	Ljung-Box Q test for autocorrelation.
LM	Lagrange multiplier test.
LRT	Likelihood ratio test.
NW	Newey West D test of restrictions.
PIT	Probability integral transformation test.
PPC	Probability plot correlation test.
RECURS	Structural stability test.
SF	Shapiro Francia normality test.
SW	Shapiro Wilks normality test.
THEIL	Theil decomposition.
TTEST	t test.
WALD	Wald test.
WELCH	Welch (ANOVA) test.

AD The Anderson Darling test evaluates whether a series exhibits normality; it is one of the most powerful statistics for detecting most departures from normality. It can be used for censored and non-censored data. The test is a one-sided and the null hypothesis that the series is derived from a normal population is rejected if the test statistic is greater than the critical value.

For the censored case, the p-value table can only provide approximate values. To provide accurate values, a p-value table, specific for the number of observations and degree of censoring, is generated in place based on 10,000 replications.

Example

```
1.  TEST (p) y;
      METHOD = AD;

2.  TEST (p) x ;
      METHOD = AD;
      CENSOR = cen;
```

The first example demonstrates how an uncensored vector y can be tested for normality, while the second shows how a Type 1 right censored vector *x* can be similarly tested, where the elements of *cen* take the value of unity if the element is censored.

Technical Notes

The Anderson-Darling test is considered parametric because it makes use of the specific distribution in calculating critical values.

ANOVA The analysis of variance test is a statistical test that is used to test the hypothesis as to whether the means of two or more populations are equal when you know that the variance of each population is the same. Note that it is assumed that each population is distributed normally. Unmatched samples (ie with missing values) are supported. The null hypothesis is equal means across populations Under the null hypothesis, the test statistic is distributed as F.

Example

```
TEST (p) x1 x2 x3;
   METHOD = ANOVA;
```

This example shows how an ANOVA analysis can be carried out on three variables - `x1, x2,` and `x3`.

ARCH The Engle Lagrange Multiplier arch test evaluates whether a series (typically residuals) exhibits an arch structure by regressing the squared residuals against lagged squared residuals of order p. The arch statistic is distributed χ^2 with p degrees of freedom.

Example

```
TEST res;
   METHOD = ARCH;
   ORDER = 4;
```

This example demonstrates how a vector `res` can be tested for an arch effect, using lags up to order `4`.

BARTLETT The Bartlett test is used to test for homogeneity of variances of k populations. Note that it is assumed that each population is distributed normally - this test is sensitive to departures from normality. The null hypothesis is equal variance across populations Under the null hypothesis, the Bartlett test statistic is distributed as χ^2 with $k - 1$ degrees of freedom.

Example

```
TEST (p) x1 x2 x3;
   METHOD = BARTLETT;
```

This example shows how a Bartlett analysis of variance can be carried out on three variables - `x1, x2,` and `x3`.

BKW The Belsley, Kuh and Walsh singular value decomposition (SVD) test evaluates the condition indexes and variance decomposition of a matrix of suitably scaled vectors as a test of multicollinearity.

Each vector is automatically scaled to unit length ie. $||x|| = 1$. Vectors consisting of logged data must be e-scaled to have zero mean; this is undertake for those variables listed in the `VLIST` option.

The first column of the output shows the condition index - the ratio of the largest singular value to each of the other singular values. A value greater than 30 is taken to be evidence of strong dependencies between the variables. The remaining columns show the variance decomposition matrix; coefficients estimated using these variables are considered degraded if more that 50% of the variance of two or more coefficients are associated with a single high condition index.

Example

```
TEST x1 x2 x3;
   METHOD = BKW;
   VLIST = x3;
```

This example demonstrates how to evaluate the degree of multicollinearity in the matrix consisting of the three vectors `x1`, `x2` and `x3`. `x3` is in logs, and so is specified in `VLIST`.

BP The Breusch-Pagan test evaluates whether the residuals in an estimated equation are homoscedastic by undertaking an auxiliary regression of the squared residuals against a set of explanatory variables. The BP statistic is distributed χ^2 with degrees of freedom equal to the number of explanatory variables in *vlist*.

Example

```
FRML eq1 y c x2 x3;
TEST (p) eq1;
   METHOD = BP;
   VLIST = c x2 x4;
```

This example tests for homoscedasticity of the residuals in eq1 based on the auxiliary regression of the squared residuals against the explanatory variables listed in vlist.

CHISQ Gives the probability that χ^2 takes a value greater than the value given in *vlist* for the degrees of freedom (required) specified in *order*.

Example

```
TEST (p) chi;
   METHOD = CHISQ;
```

```
ORDER = 5;
```

The p_value for the scalar, `chi`, distributed χ^2 with five degrees of freedom is evaluated.

CHOW The Chow test checks the stability of the regression coefficients in the model specified in *vlist* by estimating these coefficients over two or more subsamples, and evaluating the F-test based on the respective sum of squares. The specification for the subsamples is given in *periods*. If *periods* consists of *k* pairs of dates, then the Chow test will be carried out using *k* subsamples for the dates specified. If *periods* is a group name containing *k* discrete values, then the Chow test will be carried out using the *k* subsamples for the specified groups. If *periods* is a number, then *k* optimal break points are computed, where *k* is the number of periods specified. Following Quandt, the break points are chosen so as to maximize the likelihood. The periods are then compared two by two; thus for *k* = 4, there will be 6 comparisons. The Goldfeld-Quandt test for heteroscedasticity is also undertaken.

Example

```
SMPL 1956 1985;

1.  FRML eq1 y c x1 x2;
    TEST (p) eq1;
       METHOD = CHOW;
       PERIODS = 1956 1965 1966 1974 1975 1985;

2.  TEST (p) eq1;
       METHOD = CHOW;
       PERIODS = 3;
```

These two examples show how a standard Chow test for the stability of coefficients can be carried out. The first example specifies the three sub-periods exactly, while in the second, the program selects the three sub-periods using Quandt's methodology.

DF The test for a unit root in a time series is basically testing that the regression of y_t on y_{t-1} yields a coefficient of unity. The Dickey-Fuller (1979) statistic that is used in this analysis is the t-statistic for the lagged

variable for the regression of $y_t - y_{t-1}$ on y_{t-1}, with the inclusion of a constant and a trend term. The null hypothesis that the series has a unit root (ie. is integrated I(1)) yields a coefficient of zero for y_{t-1}. The t-statistic is specified in *vlist* and the number of periods is taken from the current sample unless specified in *periods*. The probability levels are evaluated based on the MacKinnon (1990) response surface estimates, on the assumption that there is both a constant and a trend term, with interpolation for intermediate values

Example

```
SMPL 1950 1990;
GENR  ylag = lag(y,1);
GENR  dely = y - ylag;
GENR  trend = numdate(_ID);
SMPL 1951 1990;
OLS dely c trend ylag;
dfstat = tstat[3];
TEST (p) dfstat;
   METHOD = DF;
```

This example shows how the Dickey-Fuller test can be used to test for unit roots. A trend variable is created using the NUMDATE command, and a OLS is carried out as shown. DFSTAT, which is the t-statistic for ylag, become the argument for the DF test.

EG The test for cointegration between time series is basically testing first that each is I(1) (ie. has unit roots) using the DF option under TEST, and second that the linear combination is I(0). This can be evaluated based on the Engle-Granger (1987) procedure. One time series is regressed against all the others, with the inclusion of a constant and a trend term. Using the residuals (ϵ) from this regression, a second regression of $\epsilon_t - \epsilon_{t-1}$ on ϵ_{t-1} is carried out, along with the inclusion of a constant and a trend term. The null hypothesis that the series are cointegrated should yield a coefficient on ϵ_{t-1} of zero. The t-statistic for this coefficient is specified in *vlist*, the number of series in *order*, and the number of periods is taken from the current sample unless specified in *periods*. The probability levels are evaluated based on the MacKinnon (1990) response surface estimates, on the assumption that there is both

a constant and a trend term, with interpolation for intermediate values.

Example

```
SMPL 1950 1990;
GENR  trend = numdate(_ID);
OLS y c trend x1 x2;
FORCST res;
   MODE = resid;
GENR reslag = lag(res,1);
GENR delres = res - reslag;
SMPL 1951 1990;
OLS delres c trend reslag;
egstat = tstat[3];
TEST (p) egstat;
   METHOD = EG;
   ORDER = 3;
```

This example tests for cointegration of the three series `y, x1,` and `x2`. It is assumed that a Dickey-Fuller test has already been carried out for each of these variables, and they have each been shown to be I(1). After creating a trend variable, an OLS is carried out with one of the series as the dependent variable, and the others (and trend and an intercept) as the explanatory variables. The residuals are created from the first regression using the FORCST command, with MODE = RESID. After creating the lagged and differenced residual, a second regression is carried out. The Engle-Granger measure `egstat` is the t-statistic on the lagged residual. The test for cointegration requires that the number of variables involved (*order*) be specified . Note that this process should probably be repeated for `x1` and `x2` as the LHS variable in the first regression.

F The F statistic is evaluated as:

$$F = \frac{s_1/f_1}{s_2/f_2}$$

where s_1 and s_2 are independent χ^2 variables with f_1 and f_2 degrees of freedom respectively. s_1 and s_2 are specified in *vlist*, and the degrees of freedom (required) are specified in *order*. Gives the probability that F takes a value greater than the calculated value for the stated degrees of

freedom.

Example

```
TEST (p) s1 s2;
   METHOD = F;
   ORDER = df1 df2;
```

If `s1` and `s2` were residual sum of squares derived from two subsamples on the same regression, with `df1` and `df2` degrees of freedom respectively, then this test would carry out the Goldfeld-Quandt test for homoscedasticity.

FTEST A set of linear restrictions on the estimated coefficients can be tested using an F test. Given a set of restrictions:

$$Rb - q = 0$$

the measure

$$\frac{(Rb - q)'(RVR')^{-1}(Rb - q)}{j}$$

is distributed as F under the null hypothesis, where V is the sample covariance matrix of the estimated coefficients, b, and j is the number of restrictions. Each row of R is the coefficients in one of the restrictions - thus R will have j rows. The procedure involves estimating the equation specified in *vlist* using OLS, and then undertaking the F test. R is specified in *rlist* and q in *value*. An alternative is to specify two equations in *vlist* - the unrestricted and restricted respectively - in which case the standard F test is undertaken.

Example

```
FRML eq1 y c x1 x2 x3;
FRML eq2 y c x1;

1.  TEST (p) eq1 eq2;
       METHOD = FTEST;

2.   r = {0 0 1 0,
          0 0 0 1};
     TEST (p) eq1;
```

```
            METHOD = FTEST
            VLIST  = r;
            VALUE  = 0 0;

3.    r = {0 0 1 1};
      TEST (p) eq1;
            METHOD = FTEST
            VLIST  = r;
            VALUE  = 1;
```

These examples show hows how one would test for coefficient restrictions. Example 1 shows how one would test for zero coefficients on x2 and x3 by specifying two separate equation names in *vlist*. Example 2 does the exact same test, but requires that the restrictions be specified using the VLIST and VALUE options. The third example shows how one could test for a single restriction - in this case that the sum of the coefficients on x2 and x3 is unity.

GRANGER Granger's causality test, or more exactly, Granger's precedence test, allows a test of whether a movement in one vector (x) precedes the movement in another vector (y). It does not relate to causality in the usual sense. A series x fails to Granger cause y if, in a regression of y on lagged y and lagged x, the coefficients of the latter are zero - this can be evaluated using a standard F-Test. The number of lags is specified in ORDER. The first *order* observations are dropped.

Example

```
TEST (p) y x;
   METHOD = GRANGER;
   ORDER = 3;
```

This example shows how the variable x can be tested to see if it Granger causes y. Three lags are used. An insignificant F-Statistic (p_value > .05) implies that x fails to cause y.

HANSEN Hansen's test of overidentifying restrictions is used to test if excess orthogonality conditions are binding in the context of single equation instrumental variable estimation. When the number of instrumental

variables (orthogonality conditions) exceeds the number of parameters to be estimated, the model is overidentified. The value of the minimum distance (the quadratic form) is specified in *vlist* and the number of restrictions - the difference between the number of instruments and the number of parameters - in *order*. Under the null hypothesis, in which the overidentifying conditions are not binding, the quadratic form is distributed as χ^2 with *order* degrees of freedom.

Example

```
FRML eq1 y c x1 x2 x3;
2SLS eq1;
   INST = c x2 x3 z1 z2 z3;
TEST (p) qf;
   METHOD = HANSEN;
   ORDER = 2;
```

This example shows how the overidentifying restrictions are tested in a previous 2SLS estimation. `qf` is the value of the minimum distance, and there are two degrees of freedom (6 instruments minus 4 parameters).

HAUSMAN Hausman's specification test is a general test for testing the hypothesis of no misspecification in the model - that is the RHS variables are independent of the residuals. The procedure involves first estimating the equation specified in *vlist* using OLS, and then estimating it using 2SLS with the instruments (required) in *instlist*. A Wald test is undertaken on the difference of the coefficients.

Example

```
FRML eq1 y c x1 x2 x3;
TEST (p) eq1;
   METHOD = HAUSMAN;
   INST = c x1 z1 z2 z3;
```

This example shows how the variables `x2` and `x3` are jointly tested for independence from the residual, using the Hausman procedure.

JB The Jarque Bera test evaluates whether a series with zero mean exhibits normality based on its skewness and kurtosis. Under the null hypothesis of normality, the JB statistic is distributed χ^2 with 2 degrees

of freedom.

Example

```
TEST (p) vseries;
   METHOD = JB;
```

This example demonstrates how a vector `vseries` can be tested for normality.

JOHANSEN The Johansen Maximum Likelihood procedure allows one to determine the number of cointegrating relationships that exist amongst the stated endogenous variables. GAUSSX evaluates the maximum eigenvalue test and the trace test for a system of equations using the error correction representation (ECM) of the VAR(k) model:

$$\Delta x_t = \mu + Bz_t + \sum_{i=1}^{k-1} \Gamma_i \Delta x_{t-i} + \Pi x_{t-k}$$

In addition, GAUSSX reports β, the matrix of orthogonalized eigen vectors (the coefficients in the error correction mechanism), $\alpha = S_{0k}\beta$, and the reduced rank long run matrix $\Pi = \alpha\beta'$ for all possible ranks.

The number of cointegrating relationships is determined by evaluating the rank (r) of Π. For both the maximum eigen value test and the trace test, the number of cointegrating vectors is determined sequentially. Starting at $r = 0$, evaluate if the null hypothesis of no cointegrating vectors is rejected. If so, test the next hypothesis that there is at most one cointegrating vector ($r \leq 1$), and so on. If $r = 0$ cannot be rejected, there are no cointegrating relationships among the x_t; if $r = k$ cannot be rejected, the hypothesis that x_t is stationary cannot be rejected; if $0 < r < k$ the hypothesis of cointegration cannot be rejected, and r indicates the number of cointegrating relationships.

In the unrestricted case of no cointegrating vectors, $r = k$, and Π is evaluated using all the columns of α and β. Under cointegration, $r < k$, the reduced rank long run matrix Π is evaluated using the first r columns of α and β.

The functional form of the system is given in the equation specified in *vlist* this is similar to the VAR command, except that the error corrected representation is estimated. Each of the endogenous variables is specified using the ENDOG option, and the maximum order of the lags is given by ORDER. While all the endogenous variables are transformed to differences, the remaining variables (constant and weakly exogenous variables) are in levels. The type of analysis undertaken is given in the MODE option:

NOTREND No deterministic trends in the endogenous variables, and no trend term in the DGP - thus $\mu = 0$.

ECMTREND Linear deterministic trends in the endogenous variables, and no trend term in the DGP.

GPTREND Linear deterministic trends in the endogenous variables, and in the DGP.

The default is NOTREND if there is no constant, and DGPTREND if there is a constant in the FRML. Note that the tabulated values for the maximal eigen value and trace test are not necessarily valid if there are exogenous variables in the ECM.

Example

```
FRML eq1 lgnp lpid lcon c;
TEST (p) eq1;
   METHOD = johansen;
   ENDOG = lgnp lpid lcon;
   ORDER = 4;
```

This carries out the Johansen procedure on the ECM specified by eq1. In this example, a system with 3 endogenous variables is specified, with the order of the underlying VAR model set to 4. Since a constant is specified in the FRML, the default is trend in the variables and in the DGP.

JTEST The Davidson and MacKinnon (1981) J-test is applied to the two equations specified in *vlist*. The procedure allows one to test between two non-nested linear models. (In the nested case, this can be achieved simply by using an F-test). The J-test consists of estimating each model (M1, M2), and deriving the fitted value of the dependent variable. Then model 1 is again estimated with the fitted value from model 2 as an addi-

tional explanatory variable, and model 2 with the fitted value from model 1. The reported J-statistic for each model is the corresponding t-statistic of the fitted value.

Example

```
FRML eq1 y c x1 x2 x3;
FRML eq2 y c x1 x4 x5;
TEST (p) eq1 eq2;
   METHOD = JTEST;
```

This example shows how the J-test is carried out on the two non-nested equations eq1 and eq2.

KPSS The KPSS test assumes that one can decompose a series into the sum of a deterministic trend, a random walk, and a stationary error. Under the null hypothesis that the series is level stationary or trend stationary, the variance of the random walk component will be zero. The KPSS procedure generates a one sided Lagrange Multiplier statistic to test the variance. This is done both for level stationarity (no trend term) and trend stationarity. Autocorrelation in the series is permitted by testing for differenced stationarity - the maximum order of correlation is specified in *order*.

Example

```
TEST gnp;
   METHOD = KPSS;
   ORDER = 8;
```

This example demonstrates how a vector gnp can be tested for stationarity, using lags up to order 8.

LBQ The Ljung_Box Q test can be used to evaluates whether autocorrelation exists in a series. Two outputs are produced:

1. Autocorrelation function. This shows the sequence of correlation between members of a single stochastic process. Thus the *k*th coefficient shows the correlation between y_t and y_{t-k}. The Ljung-Box statistic is distributed χ^2 with k degrees of freedom under the null hypothesis.
2. Partial Autocorrelation function. The *k*th order partial autocorrela-

tion coefficient measures the correlation between y_t and y_{t-k} not accounted for by an AR($k-1$) process. The sequence is the Partial Autocorrelation function. Under the null hypothesis of an AR process of order k, the test statistic is distributed approximately normally.

Example

```
TEST x1 ;
   METHOD = LBQ;
   PERIODS = k;
```

The correlogram and partial autocorrelogram for `x1` are displayed, using k lags.

LM The Lagrange multiplier test is a general test for testing the restrictions imposed on a model when only the restricted model can be estimated. The model can be linear or non-linear, and can consist of a single or multiple equations. If the restrictions are valid, the slope of the likelihood function should be near zero at the restricted estimate.
The LM statistic is evaluated as:

$$\text{LM} = \partial L/\partial\theta' I^{-1} \partial L/\partial\theta$$

where $\partial L/\partial\theta$ is the slope of the log likelihood with respect to the parameter vector, θ, and I is the information matrix. These elements are evaluated at the restricted parameter values. Under the null hypothesis that the restrictions are valid, LM is distributed as χ^2 with *order* degrees of freedom, where *order* is the number of restrictions.

Example

```
FRML eq1 y = (1/gamma(rho))*(b+x)^(-rho)
          .*y^(rho-1).*exp(-y./(b+x));
FRML es1 rho == 1;
PARAM b rho;
   VALUE = 10 1;
ML eq1;
   EQCON = es1;
TEST (p) gradvec;
   METHOD = LM;
   ORDER = 1;
```

```
   VALUE = vcov;
```

This example shows how parameter restrictions can be tested for in a non-linear context, using the Lagrange multiplier test. The unrestricted model (a general gamma distribution) has a restricted form (an exponential density) when rho is restricted to unity. gradvec and vcov are the likelihood gradient and parameter covariance matrix respectively from the previous ML command. Since there is one restriction, ORDER is set to unity.

LRT The likelihood ratio test is a general test for testing the restrictions imposed on a model. The model, which can be linear or non-linear, and consist of a single or multiple equations, is first estimated without any restrictions. The model is then re-estimated with the restrictions in place. The LRT statistic is evaluated as:

$$\text{LRT} = 2(\text{LLF1} - \text{LLF2})$$

where LLF1 is the unconstrained log likelihood and LLF2 is the constrained log likelihood. LLF1 and LLF2 are specified in *vlist*, and the number of restrictions in *order*. Under the null hypothesis, LRT is distributed as χ^2 with *order* degrees of freedom.

Example

```
FRML eq1 y = a*ln(y-r) + b*x1 + c*x2;
PARAM a r b c;
NLS eq1;
llr1 = llf;
CONST b c;
   VALUE = 0 0;
NLS eq1;
llr2 = llf;
TEST (p) llr1 llr2;
   METHOD = LRT;
   ORDER = 2;
```

This example shows how parameter restrictions can be tested for in a non-linear context, using the likelihood ratio test. The unrestricted log-likelihood is stored in llr1, and the restricted in llr2. The degrees of freedom for the LRT test, given in *order*, is two since there are two re-

strictions.

NW The Newey West D statistic is evaluated as:

$$NW = (q0 - q1)$$

where $q1$ is the unconstrained value of the minimum distance estimator (quadratic form) evaluated in a 2SLS, 3SLS or GMM estimation, and $q0$ is the constrained value of the minimum distance estimator. Under the null hypothesis that the restrictions are true, NW is distributed as χ^2 with *order* degrees of freedom.

Example

```
FRML eq1 y c x1 x2 x3 x4;
FRML eq2 y c x1 x2 ;
2SLS eq1;
   INST = c x2 z1 z2 z3 z4 z5 z6;
q1 = qf;
2SLS eq2;
   INST = c x2 z1 z2 z3 z4 z5 z6;
q0 = qf;
TEST (p) q1 q0;
   METHOD = NW;
   ORDER = 2;
```

This example shows how parameter restrictions can be tested for in an instrumental variables context, using the Newey West D test. The unrestricted minimum distance (quadratic form) is stored in q1, and the restricted in q0. The degrees of freedom for the NW test, given in *order*, is two since there are two restrictions.

PIT This test is designed to test the distributional assumption of a previous survival estimation. If the distributional assumptions are correct, then the cdf (or equivalently the survival rate) is distributed uniformly.

The PIT test generates the probability plot correlation PPC of the probability integral transformation of a data set, as well as the standard error (using the delta process). Under the null hypothesis that the data has the specified distribution, the transform is distributed uniform. The p-

value is the probability that the correlation coefficient is smaller than the value shown under the null.

The critical values for the uniform distribution are derived on the assumption of known coefficients. Consequently, since the transformation is undertaken with estimated coefficients, the reported p-values, while asymptotically correct, are biased high in small samples. The PIT test reports both the actual and the 95%lower and upper bounds of the correlation coefficient with their respective p-values. Since the true p-value falls between the actual and lower bound values, the null hypothesis is rejected if the lower bound p-value is less than 0.05. The confidence level can be changed using the BOUND option.

Example

```
FRML eq0 indx = b0 +b1*arrtemp + b2*plant;
FRML eq1 llfn = lognorm(fail,indx,scale);
FRML eq2 llfn = lognorm(fail~cen,indx,scale);

1.  ML (p,i) eq0 eq1;
    TEST (p) fail;
      METHOD = PIT;

2.  ML (p,i) eq0 eq2;
    TEST (p) fail;
      METHOD = PIT;
      CENSOR = cen;
```

The first example demonstrates how an the PIT test can be used for the uncensored vector fail. After estimating the coefficients using ML, the PIT test evaluates the survival rate using the estimated coefficients, and then evaluates the probability plot correlation coefficient based on a uniform reference distribution. Thus this test will determine if the assumption of log normality can be rejected.

The second shows how a Type 1 right censored vector *fail* can be similarly tested, where the elements of *cen* take the value of unity if the element is censored. Note that the ML estimation has to be undertaken

using the censored data.

PPC This test is designed to compare a data vector with a number of different distributions, in order to ascertain the family that the sample most likely came from. The PPC test is a measure of the linearity of a probability plot. If the sample tested is actually drawn from the hypothesized distribution, then the plot will be nearly linear, and the correlation coefficient will be close to unity. The test can be used for censored and non-censored data.

The PPC test generates the probability plot correlation coefficient for the sample, evaluated against 14 reference distributions. The one tailed critical values for each distribution are reported. In addition, the optimal value of λ from the Tukey-Lambda distribution is also reported.

The following table shows the transformations used in PPC:

Distribution	Variate	Reference
Cauchy	y	Cauchy
Expon	y	Expon
Gumbel	$-y$	SEV
Laplace	y	Laplace
Levy	$\ln(y)$	ln Levy
Logistic	y	Logistic
Loglog	$\ln(y)$	Logistic
Lognorm	$\ln(y)$	Normal
Normal	y	Normal
Pareto	$\ln(y)$	Expon
Power	$-\ln(y)$	SEV
SEV	y	SEV
Uniform	y	Uniform
Weibull	$\ln(y)$	SEV

In the default, the data is evaluated against all 14 reference distributions. Specific distributions can be tested by using the MODE option in which case a Q-Q plot will be generated for each distribution if the 'g' output option is specified.

Example

```
1.  TEST (p) x;
      METHOD = PPC;

2.  TEST (p) x;
      METHOD = PPC;
      CENSOR = cen;

3.  TEST (p,g) x;
      METHOD = PPC;
      MODE = uniform normal;
```

The first example demonstrates how the PPC test can be used for the uncensored vector y, while the second shows how a Type 1 right censored vector *x* can be similarly tested, where the elements of *cen* take the value of unity if the element is censored.

The third example demonstrates how a probability plot correlation test for two distributions is carried out on a vector x. For each distribution, a Q-Q plot is produced.

Technical Notes

The critical points were derived for each distribution based on 10,000 replications, using the Hazen order statistic for plotting position. For the uniform case, the Weibull order statistic is used.

RECURS Recursive residuals are used to perform various tests of structural stability. The residuals are estimated on the equation specified in *vlist*. The following tests are carried out and p_values reported:

- CUSUM. This test is based on the cumulative sum of standardized recursive residuals. Under the null of no structural change in the parameters, the expected value of CUSUM is zero.
- CUSUMSQ. This test is based on the cumulative sum of squared recursive residuals. Like CUSUM, this tests for structural change in the parameters.
- *t*-test. Since recursive residuals under the null hypothesis of no misspecification are iid normal with zero mean, the mean divided by the standard error is distributed as *t*.

- Runs test. This is a nonparametric test to assess serial correlation. It is based on the number of runs in which the recursive residuals maintain the same sign.
- Wilcoxon. This is also a nonparametric test for serial correlation, in which the test statistic is based on the sum of the ranked differences between successive terms.
- Von Neumann. This is a test for serial correlation, which is arithmetically very similar to the Durbin-Watson test. GAUSSX uses the significance points for the modified von Neumann ratio computed by Press and Brooks (Johnston, 1984), with interpolation for intermediate values.

Neither CUSUM nor CUSUMSQ has a test statistic; rather GAUSSX plots the 95% confidence bounds in each case so that violations can be seen graphically. A number of graphic displays are possible using recursive residuals, and these can be specified using the OPLIST option. These graphs are:

- RESID: Recursive residuals.
- CUSUM: Cumulative sum of recursive residuals.
- CUSUMSQ: Cumulative sum of squared recursive residuals.
- RCOEFF: Recursive estimates of each coefficient.
- FDIFF: First difference of the recursive estimate of each coefficient.
- ALL: All the above.
- FORWARD: Forward recursive residuals only.
- BACKWARD: Backward recursive residuals only.

Example

```
FRML eq1 y c x1 x2;
TEST (p) eq1;
   METHOD = RECURS;
   OPLIST = resid cusum cusumsq;
```

This example will carry out the recursive residual tests on eq1. It will also produce graphic output showing the recursive residuals, as well as the CUSUM and CUSUMSQ tests, for both forward and backward recursive residuals.

SF The Shapiro Francia test is used to test for the normality of a series, and is powerful against a variety of alternatives. It is best for leptokurtic (heavy tailed) samples. It can be used for censored and non-censored data.

The test is a one-sided and the null hypothesis that the series is derived from a normal population is rejected if the test statistic is less than the critical value. Critical values are restricted to sample sizes of less than 5000.

Example

```
1.  TEST (p) y;
        METHOD = SF;

2.  TEST (p) x ;
        CENSOR = cen;
        METHOD = SF;
```

The first example demonstrates how an uncensored vector y can be tested for normality, while the second shows how a Type 1 right censored vector *x* can be similarly tested, where the elements of *cen* take the value of unity if the element is censored.

SW The Shapiro Wilks test is used to test for the normality of a series, and is powerful against a variety of alternatives. It is best for platykuric (thin tailed) samples. It can be used for censored and non-censored data.

The test is a one-sided and the null hypothesis that the series is derived from a normal population is rejected if the test statistic is less than the critical value. Critical values are restricted to sample sizes of less than 5000.

Example

```
1.   TEST (p) y;
        METHOD = SW;
2.   TEST (p) x;
        CENSOR = cen;
        METHOD = SW;
```

The first example demonstrates how an uncensored vector y can be tested for normality, while the second shows how a Type 1 right censored vector *x* can be similarly tested, where the elements of *cen* take the value of unity if the element is censored.

THEIL Theil's decomposition is applied to two series, specified in *vlist*. The first variable is the actual series, while the second variable is the predicted series. Output includes the MSE, Theil's inequality coefficient, and two decompositions. Weighted analysis is available by using the WEIGHT option.

Example

```
TEST (p) act pred;
   METHOD = THEIL;
```

This example shows how Theil's decomposition analysis can be carried out on two variables - act and pred.

TTEST The *t*-test is used to test the equality of a single coefficient with some specified value. The *t* statistic is evaluated as:

$$t = \left| \frac{b - \beta}{s} \right|$$

where b is observed value, β is the population mean, and s is the sample standard error. b and s are specified in *vlist*, β is specified in *value*, and the degrees of freedom (required) is specified in *order*. If no VALUE option is specified, β is set to zero. Gives the probability that $|t|$ takes a value greater than the calculated value for the stated degrees of freedom. This is thus a two-tailed test.

Example

```
TEST (p) 1.5 .4;
   METHOD = TTEST;
   VALUE = 1;
   ORDER = 15;
```

This example shows how a t-test is carried out for the observed value of a parameter (1.5) and standard error (.4), against an expected value of unity, with 15 degrees of freedom.

WALD The Wald test is a general test for testing the restrictions imposed on a model. The model, which can be linear or non-linear, and consist of a single or multiple equations, is estimated without any restrictions. The WALD statistic is evaluated as:

$$W = [c(\hat{\theta}) - q]'(\text{Var}[c(\hat{\theta}) - q])^{-1}[c(\hat{\theta}) - q]$$

where

$$c(\hat{\theta}) = q$$

is a set of restrictions imposed on the vector of parameter estimates, $\hat{\theta}$. These restrictions are evaluated at the unrestricted parameter values. Under the null hypothesis that the restriction is true, W is distributed as χ^2 with m degrees of freedom, where m is the number of restrictions. In GAUSSX, a Wald test is carried out using the ANALYZ command.

Example

```
FRML eq1 q = a0*l^a1*k^a2;
PARAM a0 a1 a2;
NLS eq1;
FRML cq1 crstst = a1+a2-1;
TEST (p,d) cq1;
   METHOD = WALD;
```

This example shows how parameter restrictions can be tested for in a non-linear context, using the Wald test. The unrestricted Cobb-Douglas production function is estimated using NLS. A test for constant returns to scale is then undertaken by using the TEST command on the constraint equation cq1, under the null hypothesis that crstst is zero. See the ANALYZ command for full details – the two forms are equivalent.

WELCH Welch's test is used to test the hypothesis as to whether the means of two or more populations are equal when you know that each population is distributed normally, but the variance of each population is different. (ANOVA assumes each population has the same variance). Unmatched samples (ie. with missing values) are supported. The null hypothesis is equal means across populations Under the null hypothesis, Welch's test statistic is distributed as F.

Example

```
TEST (p) x1 x2 x3;
   METHOD = WELCH;
```

This example shows how an ANOVA can be carried out on three variables - x1, x2, and x3 when the variance differs across population.

It is very easy for the user to program additional tests, by using the existing tests as templates – see, for example the files testx.src, test2.src and test3.src on \gsx\gaussx. Examples of TEST are given in test14.prg, test52.prg and test61.prg.

See Also COVA

References

Anderson, T. W.; Darling, D. A. (1952). "Asymptotic theory of certain goodness-of-fit criteria based on stochastic processes". *Annals of Mathematical Statistics*, Vol. 23, pp. 193-212.

Bartlett, M.S. (1937), "Properties of Sufficiency and Statistical Tests" *Proceedings of the Royal Society of London*, Series A Vol.160, pp. 268 -282.

Belsley, D., E. Kuh, and R. Welsch (1980), *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*, John Wiley and Sons, New York.

Breusch, T.S. and A.R. Pagan (1979), "A Simple Test for Heteroscedasticity and Random Coefficient Variation", *Econometrica*, Vol. 47, pp. 1287-1294.

Brown, R., J. Durbin and J. Evans (1975), "Techniques for Testing the Constancy of Regression Relationships over Time", *Journal of the Royal Statistical Society, Series B*, Vol. 37, pp. 149-172.

Chow, G.C. (1960), "Tests of equality between sets of coefficients in two linear regressions", *Econometrica*, Vol. 28, pp. 591-605.

Davidson, J. and J. MacKinnon (1981), "Several Tests for Model Specification in the Presence of Alternative Hypotheses", *Econometrica*, Vol. 49, pp. 781-793.

Dickey, D., and W. Fuller (1979), "Distribution of the Estimators for Autoregressive Time Series with a Unit Root", *Journal of the American Statistical Association*, Vol. 74, pp. 427-431.

Engle, R. (1982), "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflations", *Econometrica*, Vol. 50, pp. 987-1008.

Engle, R., and C. Granger (1987), "Co-integration and Error Correction: Representation, Estimation and Testing", *Econometrica*, Vol. 35, pp. 251-276.

Goldfeld, S., and R. Quandt (1965), "Some Tests for Homoscedasticity", *Journal of the American Statistical Association*, Vol. 60, pp. 539-547.

Goldfeld, S., and R. Quandt (1972), *Nonlinear Methods in Econometrics*, Amsterdam, North Holland.

Granger, C.W.J. (1969), "Investigating Causal Relations by Econometric Models and Cross-Spectral Methods", *Econometrica*, Vol. 37, pp.24-36.

Hansen, L.P. (1982), "Large Sample Properties of Generalized Method of Moments Estimators", *Econometrica*, Vol 50, pp. 1029-1054.

Hausman, J.A. (1978), "Specification Tests in Econometrics", *Econometrica*, Vol. 46 (6), pp. 1251-1271.

Jarque, C.M., and A.K. Bera (1980), "Efficient tests for normality, homoscedasticity and serial independence of regression residuals", *Economic Letters*, Vol. 6, pp. 255-259.

Johansen, S. (1988), "Statistical Analysis of Cointegration Vectors", *Journal of Economic Dynamics and Control*, Vol. 12, pp. 231-254.

Johnston, J. (1984), *Econometric Methods*, 3rd ed., McGraw Hill, New York.

Kwiatowski, D., P.C.B. Phillips, P. Schmid and T. Shin (1992), "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic series have a unit root", *Journal of Econometrics*, Vol. 54, pp. 159-178.

Ljung, G. and G. Box (1979), "On a Measure of Lack of Fit in Time Series Models", *Biometrika*, Vol. 66, pp. 265-270.

Newey, W. and K. West (1987) "Hypothesid Testing with Efficient Method of Moments Estimation", *International Economic Review*, Vol 28, pp. 777-787.

Royston, P (1993), "A toolkit for testing for non-normality in complete and censored samples" *The Statistician*, Vol 42, pp. 37-43.

Theil, H. (1971), *Principles of Econometrics*, New York, Wiley.

Welch, B.L. (1951), "On the comparison of several mean values: An alternative approach." *Biometrika*, Vol. 38, pp. 330-336.

Purpose

To compute diagnostic non-parametric tests.

Format

TEST (*options*) *vlist ;*
METHOD = *methname;*
VALUE = *value;*

Input

options	optional, print options.
vlist	literal, required, variable list.
methname	literal, required, diagnostic method.
value	literal, optional, restriction values or matrix.

Remarks

The TEST command provides a number of non-parametric diagnostic tests; the test chosen is given by *methname*. Non-Parametric tests do not require specific assumptions about the distribution of the population sampled. The assumption most frequently required is that the population is continuous.

For location type tests, a single vector can be compared to a constant by specifying the single vector in *vlist*, and the constant in *value*.

The (p) option generates a pause after each screen of output. On-line help (with additional information) is available for each of these tests. The following tests are currently supported:

Location
Indep samples

KW	Kruskal-Wallis test.
MOOD	Mood's (median) test.
MW	Mann-Whitney U test.

Location
Related samples

CONOVER	Conover test.
FRIEDMAN	Friedman test.
SIGN	Sign test.
WALSH	Walsh test.
WILCOXON	Wilcoxon signed rank test.

Scale

BF	Brown-Forsythe test.
LEVENE	Levene test.
OBRIEN	O'Brien test.

Characteristics

KS	Kolmogorov-Smirnov test.
KURTOSIS	Kurtosis test.
RUNS	Runs test.
SKEWNESS	Skewness test.

BF The Brown-Forsythe test is used to test for homogeneity of variances of *k* populations. The data transformation used in the Brown-Forsythe test is:

$$y_{ij} = \left| x_{ij} - \text{median}(x_j) \right|$$

The use of the median makes the test more robust for small samples. Note that it is assumed that each population is continuous, but not necessarily normally distributed. The null hypothesis is equal variance across populations. Under the null hypothesis, the Brown-Forsythe test statistic is distributed as F.

Example

```
TEST (p) x1 x2 x3;
   METHOD = BF;
```

This example shows how a Brown-Forsythe analysis of variance can be carried out on three variables - `x1`, `x2`, and `x3`.

CONOVER The Conover test is a non-parametric test for analyzing randomized complete block designs. The concept is that there are *k* treatments applied to *n* subjects - for example *k* judges with *n* participants. The null hypothesis is that there is no difference between treatments (equal location), that is, each treatment has the same effect (no biased judges). Under the null hypothesis, the Conover test statistic is distributed as F.

Example

```
TEST (p) x1 x2 x3;
   METHOD = CONOVER;
```

This example shows how a Conover test can be used to test whether x1, x2, and x3 have the same median.

FRIEDMAN The Friedman test is a non-parametric test for analyzing randomized complete block designs. The concept is that there are k treatments applied to n subjects - for example k judges with n participants. The null hypothesis is that there is no difference between treatments (equal location), that is, each treatment has the same effect (no biased judges). Under the null hypothesis, the Friedman test statistic is distributed as χ^2 with $k - 1$ degrees of freedom.

Example

```
TEST (p) x1 x2 x3;
   METHOD = FRIEDMAN;
```

This example shows how a Friedman test can be used to test whether x1, x2, and x3 have the same median.

KS The Kolomogorov-Smirnov test is used to test whether a series comes from a specified distribution. The series must be a CDF or a survival rate. This is a nonparametric test which tests the null hypothesis that the series is indeed a cumulative distribution, and thus has a uniform distribution. Under the null hypothesis, the (corrected) Kolomogorov-Smirnov statistic is distributed normal.

Example

```
GENR sv = 1 - cdfn((y-indx)/sig);
TEST (p) sv;
   METHOD = KS;
```

This example shows how a survival rate sv is created from a series y under the assumption that y is distributed normal. This assumption can then be tested using the Kolomogorov-Smirnov test.

KW The Kruskal-Wallis test is a nonparametric test to compare three or more samples. It tests the null hypothesis that all populations have identical distribution functions against the alternative hypothesis that at least

two of the samples differ only with respect to location (median), if at all. It is the nonparametric analog to the F-test used in analysis of variance, and is a logical extension of the Mann-Whitney Test. Under the null hypothesis, the (corrected) Kruskal-Wallis statistic is distributed as χ^2 with $k - 1$ degrees of freedom.

Example

```
TEST (p) x1 x2 x3;
   METHOD = KW;
```

This example shows how a Kruskal-Wallis test can be used to test whether `x1, x2,` and `x3` have the same median.

KURTOSIS The kurtosis test is used to ascertain the value of the kurtosis of a series, and its significance. Kurtosis characterizes the relative peakedness or flatness of a distribution compared to the normal distribution. Thus the normal distribution has a kurtosis of zero, and is referred to as mesokurtic. Positive kurtosis indicates a relatively peaked distribution (leptokurtic), while negative kurtosis indicates a relatively flat distribution (platykurtic).

Under the null hypothesis that the series is mesokurtic, the kurtosis test statistic is distributed normal.

Example

```
TEST (p) x;
   METHOD = KURTOSIS;
```

This example evaluates the kurtosis of `x`, and evaluates the p_value for the null that the series is mesokurtic.

LEVENE The Levene test is used to test for homogeneity of variances of k populations. The data transformation used in the Levene test is:

$$y_{ij} = \left| x_{ij} - \text{mean}(x_j) \right|$$

Note that it is assumed that each population is continuous, but not necessarily normally distributed. The null hypothesis is equal variance across populations. Under the null hypothesis, the Levene test statistic is distributed as F.

Example

```
TEST (p) x1 x2 x3;
   METHOD = LEVENE;
```

This example shows how a Levene analysis of variance can be carried out on three variables - `x1`, `x2`, and `x3`.

MW The Mann-Whitney test (U statistic) is a ranked based non parametric test for analyzing the equality of two population medians. This test is a nonparametric alternative to the two-sample t-test - it tests whether the two population distribution functions are identical against the alternative that they differ by location. The null hypothesis is equal medians across populations.

Example

```
TEST (p) x1 x2;
   METHOD = MW;
```

This example shows how a Mann-Whitney U test can be used to test whether `x1` and `x2` have the same median.

MOOD Mood's test (or Median test) is a non-parametric test for analyzing the independence of k groups of data, based on the medians. The null hypothesis is that there is no difference between the samples - they come from a population having the same median. Under the null hypothesis, Mood's test statistic is distributed as χ^2 with $k - 1$ degrees of freedom.

Example

```
TEST (p) x1 x2 x3;
   METHOD = MOOD;
```

This example shows how Mood's test can be used to test whether `x1`, `x2`, and `x3` have the same median.

OBRIEN The O'Brien test is used to test for homogeneity of variances of k populations. It is similar to the Levene test, but is more robust. Note that it is assumed that each population is continuous, but not necessarily normally distributed. The null hypothesis is equal variance across populations. Under the null hypothesis, the O'Brien test statistic is dis-

tributed as F.

Example

```
TEST (p) x1 x2 x3;
   METHOD = OBRIEN;
```

This example shows how an O'Brien analysis of variance can be carried out on three variables - `x1, x2,` and `x3`.

RUNS The runs test is a non-parametric test for evaluating whether a single vector is non-random, by testing the order of observations in a sample. The null hypothesis is that the observations are random.

Example

```
TEST (p) x1;
   METHOD = RUNS;
```

This example shows how a test of the randomness of the elements of `x1` is carried out.

SKEWNESS The skewness test is used to ascertain the value of the skewness of a series, and its significance. Skewness characterizes the degree of asymmetry of a distribution around its mean - thus a normal distribution has a skewness of zero. Positive skewness indicates a distribution with an asymmetric tail extending towards more positive values. Negative skewness indicates a distribution with an asymmetric tail extending towards more negative values.

Under the null hypothesis that the series is sampled from a symmetrical distribution, the skewness test statistic is distributed normal.

Example

```
TEST (p) x1;
   METHOD = SKEWNESS;
```

This example evaluates the skewness of `x`, and evaluates the p_value for the null that the series is symmetrically distributed.

SIGN The Sign test is used to test the difference between the medians of two populations. Since the differences between each pair of observation is used, the test is particularly appropriate for matched pairs, where

the pairs are observed under widely different conditions. It can also be used for ordinal data. The null hypothesis is that the medians of two samples are equal. Under the null hypothesis, the Sign test statistic has a binomial distribution.

Example

```
TEST (p) x1 x2;
   METHOD = SIGN;
```

This example shows how a Sign test can be used to test whether `x1` and `x2` have the same median.

WALSH The Walsh Average (Modified Wilcoxon) test is used to test the difference between the median of two populations. Since the differences between each pair of observation is used, the test is particularly appropriate for matched pairs, where the pairs are observed under widely different conditions. The test is similar to the Wilcoxon Signed Rank test, but uses averages of pairs of differences. The null hypothesis is that the medians of two samples are equal.

Example

```
TEST (p) x1 x2;
   METHOD = WALSH;
```

This example shows how a Walsh test can be used to test whether `x1` and `x2` have the same median.

WILCOXON The Wilcoxon Signed Rank test is used to test the difference between the median of two populations. Since the differences between each pair of observation is used, the test is particularly appropriate for matched pairs, where the pairs are observed under widely different conditions. This test can also be applied when the observations in a sample of data are ranks, that is, ordinal data rather than direct measurements. The Wilcoxon Signed Rank test is more powerful than the Sign test. The null hypothesis is that the medians of two samples are equal - that is the median of paired differences is zero.

Example

```
TEST (p) x1 x2;
```

```
METHOD = WILCOXON;
```

This example shows how a Wilcoxon signed rank test can be used to test whether `x1` and `x2` have the same median.

A set of examples for non-parametric tests is given in test52.prg.

References

Brown, M.B. and A.B. Forsythe (1974), "Robust Tests for the Equality of Variances". *JASA*, Vol. 69, pp. 364-367.

Conover, W.J. (1980), *Practical Nonparametric Statistics*, Second Edition, New York, John Wiley & Sons, Inc.

Friedman, M. (1937), "The use of ranks to avoid the assumption of normality implicit in the analysis of variance", *JASA*, Vol. 32, pp. 675-701.

Kruskal, W.H. and W.A. Wallis (1952), "Use of ranks in one-criterion variance analysis". *JASA* Vol.47, pp. 583-621.

Levene, H. (1960), "Robust Tests for the Equality of Variance", *Contributions to Probability and Statistics*, ed. I. Olkin, Palo Alto, CA. Stanford University Press, pp 278 -292.

Mann, H.B., and D.R. Whitney (1947), "On a test of whether one of two random variables is stochastically larger than the other" *Annals of Mathematical Statistics*, Vol. 18, pp. 50-60.

Mood, A.M. (1954), *Introduction to the theory of statistics*. New York: McGraw Hill.

O'Brien, R.G. (1979), "A General ANOVA Method for Robust Tests of Additive Models for Variances", *JASA* Vol. 74, pp. 877-880.

Walsh, J.E. (1949), "Application of some significance tests for the median which are valid under very general conditions", *JASA*, Vol. 44, pp. 342-355.

Wilcoxon, F (1945), "Individual Comparisons by Ranking Methods", *Biometrics*, Vol 1, pp. 80-83.

Purpose

Creates a vector of log likelihoods for a truncated GARCH process (GJR).

Format

z = TGARCH (*resid, avec, bvec, gvec*);
z = TGARCH_T (*resid, avec, bvec, gvec, dvec*);

Input

resid	literal, vector of residuals.
avec	literal, vector of parameters for the ARCH process.
bvec	literal, vector of parameters for the GARCH process.
gvec	literal, vector of γ parameters.
dvec	literal, distributional parameter (ν).

Output

z	Vector of log likelihoods.
_ht	Vector of conditional variance.

Remarks

The structural coefficients and the coefficients of the TGARCH process are estimated using maximum likelihood. The TGARCH or GJR model is given by:

$$\begin{aligned} y_t &= f(x_t, \theta) + \epsilon_t \\ \epsilon_t &\sim N(0, h_t) \\ \lambda_{it} &= \gamma_i \text{ if and only if } \epsilon_{t-i} < 0 \\ h_t &= \alpha_0 + \sum_{i=1}(\alpha_i + \lambda_{it})\epsilon_{t-i}^2 + \sum_{j=1}\beta_j h_{t-j} \end{aligned}$$

The first equation describes the structural part of the model; thus this can be used for linear or non-linear structural models. The second equation specifies the distribution of the residuals, and the third equation specifies the structural form of the conditional variance h_t. The α are the vectors of the weights for the lagged ϵ^2 terms; this is the ARCH process. The β are the weights for the lagged h terms; this is the GARCH process.

avec is a vector of parameters giving the weights for the lagged asymmetric squared residuals. The first element, which is required, gives the constant. *gvec* is a vector of parameters for the asymmetric process - the order of *gvec* should be one less than the order of *avec*. *bvec* is the vector of parameters for the GARCH process. Note the stationarity conditions described under GARCH.

See the "General Notes for GARCH" under GARCH, and the "General Notes for Non-Linear Models" under NLS.

Example

```
OLS y c x1 x2;
sigsq = serˆ2;
PARAM c0 c1 c2;
   VALUE = coeff;
PARAM a0 a1 a2 b1 g1 g2;
   VALUE =  sigsq .1 .1 .1 .1 .1;
FRML cs1 a0 >= .000001;
FRML cs2 a1 >= 0;
FRML cs3 a2 >= 0;
FRML cs4 b1 >= 0;
FRML cs5 a1+a2+b1 <= .999999;
FRML eq1 resid = y - (c0 + c1*x1 + c2*x2);
FRML eq2 lf = tgarch(resid,a0|a1|a2,b1,g1|g2);
ML (p,d,i) eq1 eq2;
   EQCON = cs1 cs2 cs3 cs4 cs5;
```

In this example, a linear TGARCH model is estimated using constrained maximum likelihood, with OLS starting values. The residuals are specified in eq1, and the log likelihood is returned from eq2. Note the parameter restrictions to ensure that the variance remains positive.

Source GARCHX.SRC

See Also GARCH, EQCON, FRML, ML, NLS

References Glosten, L.R., R. Jagannathan, and D.E. Runkle (1993). "On the Relation between Expected Value and Volatility of the Nominal Excess Returns on Stocks", *Journal of Finance*, Vol 48, pp. 1779-1801.

Purpose Creates a timer control for GAUSS.

Format TIMER (*&f, param, intval, maxit*);

Input

&f	pointer to a function.
param	argument to function f.
intval	scalar, interval in milliseconds.
maxit	scalar, maximum number of iterations.

Remarks The TIMER control calls (polls) the function specified in *f* every *intval* milliseconds; thus it can be used in signal processing, or for simulating real time processes. *intval* should be longer that the time necessary to execute *f*. An optional initial parameter can be passed to *f* using *param*. The maximum number of iterations is specified in *maxit*.

The timer can be stopped at any time by typing the ESC key.

TIMER is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx;
x = 2;
timer(&myfun,3,2000,5);
print x: x;

proc 0 = myfun(ix);
  x = x+ix;
endp;
```

This example shows a simple cumulative addition. The timer calls myfun every 2000 milliseconds for a total of 5 iterations. At each iteration, param, which takes the value of 3, is added to the current value of $\hat{x}$. Thus, after 10 seconds, the program prints out the current value of x, which will be 17.

Source GXPROCS.SRC

Purpose

Creates a heading for a GAUSSX estimation procedure.

Format

GAUSSX COMMAND *vlist;*
TITLE = *title;*

Input

title string, optional, title.

Remarks

The TITLE option prints the string *title* at the beginning of most GAUSSX procedures which produce output. The string should be surrounded by quotes. The command only holds for the current command. *title* should not exceed 60 characters, otherwise some of the title will be truncated.

Example

```
OLS y c x1 x2;
   TITLE = First Regression -- x1 and x2;
```

The title "First Regression – x1 and x2" is printed at the beginning of the regression.

See Also

AR, ARCH, ARIMA, COVA, EXSMOOTH, FIML, GMM, GRAPH, KALMAN, NLS, PLOT, NPR, PANEL, POISSON, QR, ROBUST, SURE, VAR, 2SLS, 3SLS

Purpose

Creates a vector of log likelihoods for a Tobit model.

Format

z = TOBIT (*y, indx, sigma*);

Input

y	literal, dependent variable.
x	literal, index of independent variables.
sigma	literal, residual variance.

Output

z	Vector of log likelihoods.

Remarks

The Tobit coefficients are estimated using maximum likelihood; thus this can be used for linear or non-linear models. Given the unobserved latent variable y^*, and the observed variable y, then the Tobit model is given by:

$$y^* = f(x,\beta) + \epsilon$$

$$y = 0 \quad \text{if} \quad y^* \leq 0$$
$$y = y^* \quad \text{if} \quad y^* > 0$$

The dependent variable is treated as zero if y takes non positive values. Models with upper or lower truncation points at values different from zero can be estimated by an appropriate transformation of the dependent variable, or by customizing the likelihood function.

See the “General Notes for Non-Linear Models” under NLS, and the example under ML. An example is given in test09.prg.

Example

```
1.   OLS y c x1 x2;
     PARAM a0 a1 a2;
        VALUE = coeff;
     PARAM sigma;
        VALUE = ser;
     FRML eq1 indx = a0 + a1*x1 + a2*x2;
     FRML eq2 lf = TOBIT(y,indx,sigma);
     ML (p,d,i) eq1 eq2;
     GENR y = y - 5;
     ML (p,d,i) eq1 eq2;

2.   PARAM a b1 b2 sigma;
        VALUE = 1 .5 .5 1;
     FRML eq1 qhat = a*(K^b1).*(L^b2);
     FRML eq2 lf = TOBIT(q,qhat,sigma);
     ML (p,d,i) eq1 eq2;
```

In the first example, a standard linear Tobit model with truncation below zero is estimated, using OLS starting values. The RHS index is stipulated in eq1, and the log likelihood is returned from eq2. Then the Tobit model is re-estimated, with truncation below 5. Note that the constant is now biased - an unbiased constant would be 5 larger.

The second example shows how a non-linear Tobit estimation would be carried out.

Source GXPROCS.SRC

See Also ML, NLS

References Tobin, J. (1958), "Estimation of Relationships for Limited Dependent Variables", *Econometrica*, Vol. 26, pp. 24-36.

Purpose Control over trust region processing.

Format
GAUSSX COMMAND *vlist;*
STEP = *TR ;*
TRUST = *controllist;*

Input

vlist	literal, required, variable list.
controllist	literal, optional, list of control options.

Remarks Trust region methods in optimization define a region around the current iterate, and then choose the step to be the approximate minimizer of the quadratic model in this trust region. The trust region is $||\triangle b|| \leq \delta$, where $||\triangle b||$ is the norm of the change in the parameter estimate, and δ is a scalar defining the size of the trust region. If the prediction given by the change in the quadratic model is close to that of the actual function change, then the trust region is increased, i.e. δ is increased. On the other hand, if the quadratic approximation is poor, δ will be decreased.

The trust region methodology is implemented for unconstrained non-linear optimization (NLS, FIML, GMM, ML) when the STEP type is set to TR.

Control over the trust region options is provided by the TRUST option; this consists of a 4 element vector *controllist*; these elements are:

1. Initial size of region (s). δ is then calculated as $\sqrt{ks^2}$, where k is the number of parameters. Default = 0.1.
2. Maximum size of region (m). ($s \leq m$). δ is then calculated as $\sqrt{km^2}$. Default = 1.
3. Tolerance for Newton estimate of Lagrange multiplier. Default = 0.001.
4. Maximum number of Newton iterations. Default = 3.

For a problem with a large number of parameters, creating the Hessian is time consuming. For the trust region method, a quasi-Newton methodology (eg. BFGS) can create an initial step direction and find the optimum position in the trust region without having to do a Cholesky factorization - thus this can be an efficient method for large problems.

The trust region step methodology is especially useful for hard problems. In econometrics, finding reasonable starting values for parameters is often difficult, and poor starting values often result in a "*Failure to Improve Objective Function*" error. Trust region step methodology can often provide a resolution to this type of problem.

An example of trust region optimization is given in test28.prg - this example uses a 20 parameter Rosenbrock function, with starting values at $x(i) = 200$, a long way from the optimum $x(i) = 200$.

Example

```
ML (p,i) eq1;
   STEP = tr;
   TRUST =  .5 2 .001 5;
   METHOD = nr nr nr;
```

This example would undertake maximum likelihood on eq1 using trust region step method.

See Also FIML, GMM, ML, NLS

References Nocedal, J. and S Wright,(1999). *Numerical Optimization* Springer, New York.

Purpose

Estimates the coefficients of vector autoregressive system of equations.

Format

VAR (*options*) *elist ;*
ENDOG = *endlist;*
METHOD = *methname;*
ORDER = *lags;*
PDL = *pdllist;*
PERIODS = *irfnum;*
TITLE = *title;*
WEIGHT = *wtname;*
WINDOW = *windowtype;*

Input

options	optional, print options.
elist	literal, required, variable list or equation name.
endlist	literal, optional, endogenous variable list.
methname	literal, optional, covariance method (NONE).
lags	numeric, optional, number of lags used (1).
pdllist	literal, optional, options for PDL.
irfnum	numeric, optional, number of periods used by IRF (0).
title	string, optional, title.
wtname	literal, optional, weighting variable.
windowtype	literal/numeric, optional, spectral window.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
COVU	Residual Covariance matrix.

Remarks

The VAR command estimates a system of equations, in which the lagged values of the endogenous variables occur on the RHS. Since lagged variables are to be created from the endogenous variable names, the names should be short (5 characters max.) to allow for the lagged component. GAUSSX uses the current sample to estimate the VAR process, dropping the first *lags* cases

to allow for the lagged variables. Each equation has exactly the same terms on the RHS (and can thus be efficiently estimated using OLS).

Optionally, the impulse response function (IRF) and the forecast error decomposition (FED) are available. The user specifies the endogenous and exogenous variables of the system in a FRML, as well as the maximum order of the lags to be used for the endogenous variables. If the IRF is to be produced, the user must also specify the number of periods to be used in tracing the IRF. The IRF and FED will change depending on the ordering of the endogenous variables, since residual covariance is assigned to the equation that comes first. Further information on the IRF and FED is given in the on line help.

See the "General Notes for Linear Models" under OLS. An example is given in test02.prg, and a Johansen test for cointegration in test19.prg.

Example

```
1.   VAR (p) y c x1 x2 ;

2.   FRML eq1 y m1 c x1 x2 x3;
     VAR (p,d,s,v) eq1;
        ENDOG   = y m1;
        ORDER   = 2;
        PERIODS = 6;
```

In the first example, a single equation VAR is estimated - since no subcommands are used, the default is a single equation, with the first variable specified being the endogenous variable, and a lag structure of order one. Thus y is regressed on c, x1, x2, and y(-1). The screen display pauses (p) after each screen.

The second example shows how a two equation system is specified. The FRML specifies the entire list of endogenous and exogenous variables of the system. The list of endogenous variables is specified by the ENDOG option. A lag structure of order 2 is used; thus the RHS variables for each equation will consist of y(-1), y(-2,) m1(-1), m1(-2), c, x1, x2 and x3. The PERIODS option will generate both the impulse response function and the forecast error decomposition for 6 periods. The screen pauses (p) after each

display, and descriptive statistics (d) are printed. Both the variance covariance matrix (v) and equation diagnostic statistics (s) are produced for each equation.

See Also AR, FRML, OLS, PDL, VAR, WEIGHT, WINDOW

References Box, G., and G. Tiao (1981), "Modelling Multiple Times Series with Applications", *Journal of the American Statistical Association*, Vol. 76, pp. 802-816.

Purpose

Creates a matrix of fitted values for a vector autoregressive moving average process.

Format

z = VARMA (*y, phi, theta*);
OPLIST = *progopts ;*

Input

y	literal, *N*x*K* matrix of time series.
phi	literal, (*P***K*)x*K* AR coefficient matrix, or scalar zero.
theta	literal, (*Q***K*)x*K* MA coefficient matrix, or scalar zero.
progopts	literal, optional program options

Output

z *N*x*K* matrix of fitted values.

Remarks

The coefficients of the VARMA process are estimated using NLS. When there is no MA component, this becomes the VAR model. When only a single time series is specified, this becomes the ARMA model.

The program control options are specified in *oplist*. The options available are:

`CONSTANT/[NOCONST]` Specifies whether a constant is to be included. CONSTANT should normally be specified for non-differenced series with non-zero mean, unless the constant is explicitly specified as a parameter.

Both stationary and invertibility conditions need to be satisfied. GAUSSX provides a routine called `mroot`, which returns the value of the largest root, which must have a modulus less than unity. Consequently, constrained NLS is usually required.

An example of VARMA is given in test39.prg.

See the “General Notes for Non-Linear Models” under NLS.

Example

```
FRML eqw w :=  varma(y1~y2, p_mat, q_mat);
FRML eq21 y1 = c1 + submat(w,0,1);
FRML eq22 y2 = c2 + submat(w,0,2);
FRML ec1 mroot(p_mat) <= .9999;
FRML ec2 mroot(q_mat) <= .9999;

NLS (p,d,i) eq21 eq22;
   EQSUB  = eqw;
   EQCON = ec1 ec2;
```

In this example, eqw returns the matrix of fitted values based on AR coefficient matrix p_mat and MA coefficient matrix q_mat. These are estimated using constrained NLS, where the constraints are specified in ec1 and ec2, and where mroot is a GAUSSX routine for returning the value of the largest root in the complex plane.

Source VARMAX.SRC

See Also ARIMA, ARMA, MROOT, NLS, VAR

References Hamilton, J.D. (1994), *Time Series Analysis*, Ch. 11.

Purpose

Creates a vector of log likelihoods for a Weibull process.

Format

z = WEIBULL (*y, indx, pvec*);

Input

y	literal, dependent variable - duration.
indx	literal, scale index
pvec	literal, shape parameter

Output

z	Vector of log likelihoods.

Remarks

The Weibull model can be used to estimate duration data. The expected value of $scale_i$ is parameterized as:

$$E(scale_i) = \exp(indx_i).$$

where the index is a function of explanatory variables, x_i:

$$indx_i = f(x_i, \beta)$$

The coefficients, β and *pvec* are estimated using maximum likelihood; thus this can be used for linear or non-linear models. In the Weibull distribution, scale is the characteristic life. *shape* is the positive shape parameter.

In the default, there is no censoring. Censoring occurs if units are removed prior to failure, or are still operating at the conclusion of the test (right censored). For the censored case, *y* is an *N*x2 matrix, with the first column being the duration value, and each element of the second column taking a value of unity if the unit was censored, else zero.

See the "General Notes for Non-Linear Models" under NLS. An example is given in test57.prg.

Example

```
   PARAM b0 b1 b2 shape;
   FRML eq0 scale = b0 +b1*arrtemp + b2*plant;
   FRML ec1 shape >= 0;

1  FRML eq1 llfn =  weibull(fail,scale,shape);
   ML (p,i) eq0 eq1;
      EQCON = ec1;

2  FRML eq2 llfn =  weibull(fail~censor,scale,shape);
   ML (p,i) eq0 eq2;
      EQCON = ec1;
```

In example 1, a Weibull model is estimated using constrained maximum likelihood, with the scale index defined in `eq0`, and the log likelihood in `eq1`.

Example 2 shows a similar estimation when some of the data is censored.

Source DURATION.SRC

See Also DURATION, ML, NLS

Purpose

To specify the weights for weighted analysis.

Format

GAUSSX COMMAND *vlist;*
WEIGHT = *wtname;*

Input

vlist literal, required, variable list.
wtname literal, required, weighting variable.

Remarks

Weighted regressions are available using the WEIGHT option. This option is available for all parametric estimation models Negative weights are given zero value, and weights are normalized such that the sum of the weights equals the number of observations.

For the descriptive statistics (COVA) and the linear regression models, each observation is multiplied by square root of the normalized weight, and the statistics and diagnostics generated are for the weighted data. For GMM, NLS and FIML, the residuals are multiplied by the square-root of the normalized weights in a manner similar to OLS. For the probability models (ML, QR, and POISSON models), the log of the likelihood for each observation is weighted by the normalized weight.

Weighted least squares is used when the variance of the disturbances in a regression are known to differ across observations; in this context, it is equivalent to generalized least squares. The series *wtname* should be proportional to the inverse of the variances of the disturbances in the regression. For cross-section data, the weight should equal population if the dependent variable is per-capita, and should equal the reciprocal of population if the dependent variable is an aggregate.

Example

```
OLS y c x1 x2 ;
   WEIGHT = wtvar;
```

In this example, a weighted regression is estimated using the vector `wtvar` as the weight. This is equivalent to doing GLS on a heteroscedastic model.

See Also

AR, ARCH, COVA, FIML, GMM, NLS, POISSON, QR, SURE, VAR, 2SLS, 3SLS

Purpose

Evaluates consumer surplus associated with a given change in prices.

Format

{ *cs, se, dwl* } = WELFARE (*mth, &fct, pmat, y, b, bcov*);

Input

mth	string, consumer surplus method.
&fct	literal, required, demand function procedure.
pmat	*K*x2 matrix, required, initial and final prices.
y	scalar, required, income.
b	*R*x1 vector, optional, parameters of the demand function.
bcov	*R*x*R* matrix, optional, covariance matrix of b, or scalar zero
_wStep	global scalar, number of steps (default 20).
_wPrint	global scalar, output flag: 0 - off, 1 - on. (default=1)

Output

cs	consumer surplus.
se	standard error of cs.
dwl	deadweight loss

Remarks

The WELFARE command evaluates consumer surplus and deadweight loss for a given set of price changes for a demand system. Three methods are available, and are specified in *mth*:

CV	Compensating Variation.
EV	Equivalent Variation.
MS	Marshallian Surplus.

The standard error of the computed consumer surplus and deadweight loss is evaluated if the estimated parameter covariance matrix *bcov* is specified.

The *K* equation demand system is specified in &fct. This is a pointer to a procedure that takes three input arguments:

1	*K*x2 matrix of initial and final prices.
2	Income.
3	*R*x1 vector of parameters.

An example of WELFARE is given in test46.prg.

WELFARE is pure GAUSS code, and can be used independently of GAUSSX.

Example

```
library gaussx;
  let p0 = 1.0 2.0;                  @ initial price vector   @
  let p1 = 1.5 2.5;                  @ final price vector     @
  y  = 220;                          @ income                 @
  let b = 1 1;                       @ demand function params @
  let omega[2,2] =  .10 -.05         @ var-covariance matrix  @
                   -.05  .05;        @ of estimated coeffs    @

  {cv, secv, dwl} =  welfare(cv,&qfn, p0~p1, y, b, omega);


  proc qfn(p,y,b);                   @ demand function        @
    local pa, pb, za, zb;
    pa = p[1]; pb = p[2];
    za = b[1]*pb*y/(pa*(b[2]*pa+b[1]*pb));
    zb = b[2]*pa*y/(pb*(b[2]*pa+b[1]*pb));
    retp(za|zb);
 endp;
```

This example evaluates the compensating variation and deadweight loss for a two good system with an initial price p0 and a final price p1. Typically, the parameter vector b and its association covariance matrix would be derived from a previous estimation.

Source

WELFAREX.SRC

References

Breslaw, J.A. and J.B. Smith (1995), "A simple and efficient method for estimating the magnitude and precision of welfare changes", *Journal of Applied Econometrics*, Vol 10, pp. 313-327.

Purpose Creates a vector of log likelihoods for a local Whittle process.

Format
```
z = WHITTLE ( y, d) );
    OPLIST = oplist;
    PERIODS = periods;
```

Input

y	literal, *N*x1 vector of time series.
d	scalar, degree of differencing.
oplist	literal, optional, program options.
periods	literal, optional, truncation value.

Output

z	Vector of log likelihoods.

Remarks The local Whittle estimator is a semi-parametric estimator of the degree of differencing in a fractionally integrated process, based on the periodogram.

The fractionally integrated process is given by:

$$(1 - L)^d y_t = \epsilon_t 1\{t \geq 1\}, \ t = 0, \pm 1, \ldots$$

where L is the backward shift operator, $1\{.\}$ is the indicator function and d is the fractional degree of differencing. d is estimated using maximum likelihood.

The local Whittle estimator involves the summation of the frequencies up to $2\pi m/n$ where m is the truncation value, and is specified using the PERIODS option. The default value is $n^{0.6}$

The program control options are specified in *oplist*. The options available are:

[LW]/ELW/FELW Specifies the estimation method:

LW	Local Whittle
ELW	Exact Local Whittle
FELW	Feasible Exact Local Whittle

[PAD]/NOPAD Specifies whether padding will occur for the Fourier transform. Given the sample size, the fast Fourier transform will always be used if padding is not required. Otherwise, if NOPAD is specified, the slower discrete Fourier transform will be used.

An example is given in test55.prg.

Example

```
FRML eq1 llf = WHITTLE(y, d);
ML (p,i) eq1 ;
   PERIODS = 80;
   OPLIST = elw;
```

In this example, the memory parameter d for the time series y is estimated using an exact local Whittle estimator based on a truncation value of 80.

Source

WHITTLEX.SRC

References

Robinson, P.M. (1995), "Gaussian semiparametric estimation of long range dependences", *Annals of Statistics*, Vol. 23, pp. 1630-1661.

Shimotsu, K and P.C.B Phillips (2005), "Exact Local Whittle Estimation of Fractional Integration", *Annals of Statistics*, Vol. 33, pp. 1890-1933.

Purpose

To specify a spectral window.

Format

GAUSSX COMMAND *vlist;*
WINDOW = *wintype winsize;*

Input

vlist	literal, required, variable list.
wintype	literal, optional, spectral window.
winsize	numeric, optional, window width.

Remarks

Spectral windows are used in GAUSSX both in NPR, and in evaluating parameter covariance matrices which are consistent to residual autocorrelation (Newey-West).

For NPR, only the *winsize* parameter is used, as a measure of the window width.

For Newey-West, *winsize* gives the maximum lag length, ie. the number of autocorrelation terms, while *wintype* gives the spectral kernel. The available kernels are [BARTLETT], HANNING, PARZEN, UNIFORM, and WELCH. These are defined in Press *et al*, 1986.

Example

```
1.   OLS y c x1 x2 ;
        METHOD = robust;
        WINDOW = parzen 2;

2.   NLS eqn1 eqn2;
        INST = c x1 x2 x3;
        METHOD = nr nr robust;
        WINDOW = 1;
```

In the first example, an OLS regression is estimated with a parameter Newey-West covariance matrix based on a lag length of 2, and a Parzen window.

The second example shows a non-linear 3SLS with a Newey-West covariance matrix based on a lag length of 1, and a BARTLETT (default) window.

See Also GMM, NLS, NPR, OLS, SURE, VAR, 2SLS, 3SLS

References

Press W.H.,*et al* (1986), "Numerical Recipes", Cambridge University Press, Cambridge.

Newey, W.K., and K.D. West (1987), "A Simple Positive Semi-Definite Heteroskedasticity and Autocorrelation Consistent Covariance Matrix", *Econometrica*, Vol. 55, pp. 703-708.

Purpose

Estimates the coefficients in an equation using two stage least squares.

Format

2SLS (*options*) *vlist ;*
INST = *instlist;*
METHOD = *methname;*
PDL = *pdllist;*
TITLE = *title;*
WEIGHT = *wtname;*
WINDOW = *windowtype;*

Input

options	optional, print options.
vlist	literal, required, variable list or equation name.
instlist	literal, required, list of instruments.
methname	literal, optional, covariance method (NONE).
pdllist	literal, optional, options for PDL.
title	string, optional, title.
wtname	literal, optional, weighting variable.
windowtype	literal/numeric, optional, spectral window.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
RSS	Residual sum of squares.
SER	Standard error of the regression.
FSTAT	F–statistic.
QF	Quadratic form
RSQ	R-squared.
RBARSQ	RBAR-squared.
VCOV	Parameter covariance matrix.

Remarks

The 2SLS command carries out two stage least squares. In such a regression, the user must specify a list of instrumental variablesinstrumental variables – there must be at least as many instruments as there are independent variables. Usually a constant (C) is needed in the list of instruments.

See the "General Notes for Linear Model" under OLS. Non-linear 2SLS is described under NLS. An example of 2SLS is given in test02.prg.

Example

```
1.   2SLS y c x1 x2 ;
        INST = c x1 x3 x4;

2.   2SLS (d,p,s) eq1;
        INST = c x1 x2(-1) x4;
        WEIGHT = wtvar;
```

In the first example, 2SLS is performed with y as the dependent variable, and c, x1 and x2 as the independent variables; the instruments are specified in the INST list.

In example 2, a weighted 2SLS analysis is performed on the structural equation specified in eq1, with the instruments specified in *instlist*; execution pauses (p) after each screen display, and descriptive (d) and diagnostic (s) statistics are produced.

See Also

FRML, NLS, OLS, PDL, TITLE, WEIGHT, WINDOW, 3SLS

References

Greene, W.H. (1993), *Econometric Analysis*, 2nd ed. Macmillan, New York.

Purpose

Estimates the coefficients in a system of equations using three stage least squares.

Format

3SLS (*options*) *elist ;*
INST = *instlist;*
METHOD = *methname;*
PDL = *pdllist;*
TITLE = *title;*
WEIGHT = *wtname;*
WINDOW = *windowtype;*

Input

options	optional, print options.
elist	literal, required, equation list.
instlist	literal, required, list of instruments.
methname	literal, optional, covariance method (NONE).
pdllist	literal, optional, options for PDL.
title	string, optional, title.
wtname	literal, optional, weighting variable.
windowtype	literal/numeric, optional, spectral window.

Output

COEFF	Vector of coefficients.
STDERR	Vector of standard errors.
TSTAT	Vector of t-statistics.
ETA_B	Vector of elasticities.
ETA_SE	Vector of std. error of elasticities.
ETA_T	Vector of t-stat. of elasticities.
QF	Quadratic form
VCOV	Parameter Covariance matrix.
COVU	Residual Covariance matrix.

Remarks

The 3SLS command estimates a system of linear equations (a stacked equation system) in two stages. In the first stage each equation is estimated using 2SLS; then using the estimated variance covariance matrix of residuals, the system is estimated using generalized least squares. As in 2SLS, the user must specify a list of instrumental variablesinstrumental variables—these are usually (though not always) the exogenous variables of the system. It is up to the user to make sure that each equation is identified.

See the "General Notes for Linear Model" under OLS. Non-linear 3SLS is described under NLS. An example of 3SLS is given in test02.prg.

Example

```
FRML eq1 y1 c x1 x2 y2;
FRML eq2 y2 c x1 x3 x4 y1;
FRML eq3 y2 c x1 x3 y1;

1.   3SLS (p) eq1 eq2;
        INST = c x1 x2 x3 x4;

2.   3SLS eq1 eq3;
        INST = c x1 x2 x3 ;
```

Example 1 estimates the system of linear equations specified in eq1 and eq2 using 3SLS; execution pauses (p) after each screen display.

Example 2 shows an exactly identified system; thus the coefficient estimates from 2SLS and 3SLS are identical.

See Also FRML, NLS, OLS, PDL, TITLE, WEIGHT, WINDOW, 2SLS

References Greene, W.H. (1993), *Econometric Analysis*, 2nd ed. Macmillan, New York.

Appendices A

A.1 Error Processing.

A.2 Installing New Procedures.

A.3 Mixing GAUSS and GAUSSX.

A.4 Running GAUSS Application Modules.

A.5 Automatic Differentiation.

A.6 Support Functions.

A.7 Trouble Shooting.

A.8 Statlib Distributions]

A.1 Error Processing

Error processing occurs jointly by GAUSS and GAUSSX. If an error is encountered, the usual response is that the program will be terminated with an error message. If GAUSSX processed the error, the nature of the error will be described, and the user is returned to the GAUSS prompt in Windows, and to the GAUSSX menu in UNIX. If GAUSS processes the error, the line number (in parenthesis) where the error occurred is reported, and the user will be returned to the GAUSS prompt. The user should either select the `Gaussx / Display Error File` menu item, or type "Ctrl-F3", or enter the command:

```
gaussx ←┘
```

GAUSSX will display the parsed file (gxfile.prg), with the error highlighted, so that the user can see exactly where the error occurred.

In some situations, it may be necessary to clear memory before restarting GAUSS; this can be achieved by typing at the GAUSS prompt:

```
new ←┘
```

Once the command file has been parsed, GAUSSX transfers control to GAUSS, which scans the program code and compiles it into a form that it can then execute. At this stage, all error processing will be done by GAUSS. Typical errors include incorrect syntax, and specifying variables in equations before they have been defined. For example, specifying an estimation process (eg. 2SLS) before the appropriate equations (FRML) have been specified will cause GAUSS to abort the compilation. Note that the line number reported by GAUSS in these circumstances refer to the gxfile.prg file.

Once execution commences, errors can be trapped by either GAUSS or GAUSSX. The GAUSSX error processor will identify the current command, the current subroutine, and the nature of the problem. The narrative in these cases will indicate what corrections are necessary. In some cases a warning will be given; again, the nature of the problem will be clear.

Errors not trapped by GAUSSX will be trapped by GAUSS. In these cases, a message will be printed, and a line number of the procedure in which the error occurred will also be printed if the `LINES ON` option is specified. These types of errors will usually occur because of an incorrect GAUSS operation by the user. For example, if, in an equation, the user specified a

matrix operation between non-conformable matrices, than this would result in a the error being trapped by GAUSS. GAUSSX errors would also be trapped in this way. Some indication of the nature of the problem can usually be ascertained by asking why this particular command failed, while similar commands were successfully executed. For example, a 2SLS might work just fine, yet a subsequent 2SLS might fail because the user specified the dependent variable amongst the list of instruments. If all else fails, look at the source coding for the procedure mentioned on gsx\gaussx, and read the chapter on "Error Handling" in the GAUSS manual for more details.

A.2 Installing New Procedures

The files in the folder \gauss\gsx\gaussx contains the The entire source code for the compiled GAUSSX routines. This means that it is relatively simple for the user to make changes to the way a particular procedure runs, or to write new procedures. The location of non-compiled GAUSS code is specified under the appropriate command, and will be on the \gauss\src directory.

To make a change to an existing procedure, you need to first find the appropriate files. Look first at the file init.src. You will see a variable called "_table", whose first column contains all the GAUSSX commands. The corresponding element of the second column is the GAUSSX call for that command. For example, the second column for the command OLS is SYSX. SYSX will typically be found in the file with the same name – sysx.src. Now, edit this file from GAUSS:

```
edit \gauss\gsx\gaussx\sysx.src
```

make your change to this file (make sure you have a backup), and then execute it (Ctrl-F2). Next time GAUSSX runs, it will load the new version of this compiled file.

Experienced GAUSS users who find that they prefer to run GAUSS under the GAUSSX environment may wish to install statistical procedures of their own. This appendix describes the steps necessary to undertake this process, and in so doing describes the operation of GAUSSX is some detail. This is somewhat technical, so don't try unless you are familiar with both GAUSS and GAUSSX.

Start off with a file which is not dissimilar to the type of process that you wish to program. For example, a simple estimation might use the arx.src file as the template; a descriptive procedure might use covax.src as its template. Follow the basic style - the way data is read in is fairly standard. The arguments of each GAUSSX command are stored in a vector called "_data". Other options (MAXIT, etc.) can be used as is. You may need a couple of procedures to do the full job. Compile these procedures, and save them to the \CP subdirectory. Now, having done the programming, you need to incorporate the new procedures into GAUSSX . First, add these procedures as dummy procedures within the file src\gaussx.prc, and augment "_arglst" in the same file. Next, add the command to the matrix "_table" in `init.src`, and recompile that file; don't forget to augment the size of "_table". Now, go to the file `src\gaussx2.src`, and add the necessary coding to the proc. EXEGX. To run this new version, you need to create a new gaussxcp.gcg file; run the file \gauss\gsx\compile.prg from GAUSS. Finally, you must test your new command, to make sure it operates correctly.

A.3 Mixing GAUSS and GAUSSX

This appendix describes howGAUSSX can be used with GAUSS to carry out tasks that are not possible with just GAUSSX.

GAUSSX variables – those that are available from opening a data set, or created with a GENR, LOAD, STORE, FORCST or SOLVE command, – are stored in the GAUSSX workspace. After each command has been executed, each of these variables is set to scalar zero. This clears out the GAUSS workspace, which otherwise would get filled up very quickly. Thus, in general, a GAUSS statement on a GAUSSX variable will fail unless that variable has been copied into the GAUSS workspace using the FETCH command. Note that when a GAUSS variable is copied to the GAUSSX workspace using the STORE command, it becomes a GAUSSX variable, and will be set to scalar zero. Note also that the default for GAUSSX is OPTION = SPACE. Setting OPTION = TIME does not clear out data and procedures; as such, it is especially useful in a DO loop.

Program control is best handled using GAUSS commands. DO..ENDO work exactly as described in the GAUSS manual, and can include a block of GAUSSX commands. IF..ELSE..ENDIF works similarly. GOTO requires that the label is a separate command:

```
GOTO FINISH;
.
.
FINISH:;
```

One of the most powerful tools in GAUSSX is the ML command. However, it is often the case that the likelihood cannot be built up using a series of FRMLs. The solution is to specify a PROC within a FRML:

```
PARAM a1 a2 a3;
FRML eq1 llfn = userprc(x1~x2~x3,a1~a2~a3);
ML eq1;
PROC userprc(x,a);
.
.
retp(llf);
endp;
```

Note that the variables and parameters must be specified, either in the likelihood FRML, or in a dummy FRML. Userprc must return a vector of log likelihoods. See the coding on src\gxprocs.src for examples. Frequently used procedures can be added to gxprocs.src; the item must be added to the GAUSSX library \gauss\lib\gaussx.lcg.

A.4 Running GAUSS Application Modules

Many of the GAUSS Application Modules can be run from GAUSSX. Since the format of the Application Modules is quite standardized, it makes sense to use these modules as an extension of the GAUSSX language. As an example, suppose that one wanted to list the variables in a GAUSSX dataset. This can be done very simply using the Data Utilities Module:

```
create 1 1000;
library dutil;
open; fname = myset;
@@ call ddesc(dataset);
end;
```

Note in this example that the variable dataset has already been initialized by GAUSSX .

As another example, consider using the TTEST command in the Basic Statistic Module for testing the differences of means between two groups:

```
create 1 100;
library bstat;
open; fname = myset;
bstatset;
let vlist = x1 x2;
let grpvar = size;
_ttcut = 2 ;
@@ call ttest(dataset,grpvar,vars);
end;
```

This is a standard GAUSSX command file, interspersed with GAUSS statements. The library command is necessary to inform GAUSS where to find the subsequent files. The error UNDEFINED SYMBOL: TTEST results if this command is not included. All the variables, such as vlist, as well as any globals, such as _ttcut, must be defined prior to the procedure call. The @@ symbol forces output ON so that the resulting output is written to the file specified on the GAUSSX menu. dataset is a GAUSSX reserved word that contains the name of the current GAUSSX workspace - usually TEMP1 or TEMP2. The default selection is:

```
_dtsel = {_sample eq 1};
```

This ensures that the current SMPL is respected.

There is a significant difference between a GAUSSX command and an Application Module. The former is loaded in the compiled form, executed, and then cleared from memory. The latter is first compiled from source coding, which results in a significant delay prior to execution. After execution, the modules remain resident. Thus it is easy to get workspace problems. If a module cannot load because of insufficient memory change the value of "__rowfac"; GAUSSX uses an initial value of 0.5. To clear out memory after using an application module, use:

```
new; ←┘
```

A.5 Automatic Differentiation

The GAUSSX default mode of estimating gradients and Hessians for non-linear optimization is to use finite differencing - for example:

$$dsin/dx = (sin(x + h) - sin(x))/h$$

This is a numerical solution, and while it is easy to implement, it is slow. especially when estimating the Hessian with a large number of parameters. It can also be inaccurate. An analytic solution:

$$dsin/dx = cos(x)$$

is much faster and more accurate. If Maple 9 or 9.5 is installed, GAUSSX can use the Maple kernel to generate automatic differentiation (AD) code as GAUSS procedures. Maple is distributed by Waterloo Maple, Inc, Canada.

AD under Gaussx requires minimal work by the user. The command

```
OPTION ad;
```

will result in all subsequent gradients and Hessians to be evaluated analytically. Alternatively, individual analytic gradients and/or Hessians can be invoked using the existing syntax:

```
gradient = &symgrad;
hessian = &symhess;
```

When GAUSSX encounters the keywords `&symgrad` or `&symhess`, it loads the Maple kernel, processes the FRML that are used in the current estimation, and creates GAUSS procedures `symgrad` and/or `symhess` which return the respective gradients and/or Hessian, using the symbolic processing capability of Maple. These procedures can be saved using the SAVEPROC command, and loaded using the LOADPROC command.

AD works with ML, NLS and FIML. Depending of the size and type of problem analytic gradients and Hessians can result in speed increases for optimization problems of between 3 and 10 fold. Note that the Hessian is only used if the `NR` algorithm is specified.

AD can be used for most of the estimation processes specified in GAUSSX. For many processes, such as MNL, TOBIT and GARCH, the gradients are evaluated based on AD applied to

the entire code. For other processes - such as MNP and SV - the gradients are evaluated using a mixture of analytic and numeric techniques.

Unless an error occurs, the process is completely transparent to the user. A full diagnostic evaluation of the process is available by specifying "s" as part of the print option.

Example

```
ML (p,i,s) eq1 eq2;
    METHOD = nr bhhh nr;
    GRADIENT = &symgrad;
```

In this example, gradients are evaluated using automatic differentiation. The print option "s" generates a complete diagnostic.

An example is shown in test42.prg. The timings shown exclude the overhead of creating the AD procedures in Maple. Use of the SAVEPROC and LOADPROC commands eliminates this overhead.

A.6 Support Functions

While the reference section provides details of GAUSSX commands, there are a number of other support commands that provide the required functionality. For example, the ability to forecast an ARFIMA process requires the autocovariance function for this process, and this proc (acv) is available. This appendix lists with a brief description these support functions.

ACF

r = ACF (*d, phi, theta, n*);

Computes autocorrelation function of an ARMA (d = 0) or ARFIMA ($d < .5$) given difference parameter, *d*, AR coefficients *phi*, and MA coefficients *theta*. *r* is the *n*x1 vector of autocorrelation coefficients.

ACV

v = ACV (*d, phi, theta, n*);

Computes autocovariance function of an ARMA (d = 0) or ARFIMA ($d < .5$) given difference parameter, *d*, AR coefficients *phi*, and MA coefficients *theta*. *v* is the *n*x1 vector of autocovariance coefficients.

ARCCOSH

py = ARCCOSH (*x*);

The function *arccosh*(*x*) is the inverse function of the function cosh(*x*).

```
y = argcosh(x);
```

ARCSINH

py = ARCSINH (*x*);

The function *arcsinh*(*x*) is the inverse function of the function sinh(*x*).

```
y = argsinh(x);
```

ARCTANH

py = ARCTANH (*x*);

The function *arctanh*(*x*) is the inverse function of the function tanh(*x*).

```
y = argtanh(x);
```

CENMEANC *mu* = CENMEANC (*x, cen, ctype*);

The function returns the mean for a type 1 censored sample from a normal population, where *x* is the sample vector, *cen* is the censor vector with elements of unity for censored observations, and *ctype* is set to zero for left censored, and unity for right censored.

```
y = cenmeanc(x,cen,1);
```

CENSTDC *mu* = CENSTDC (*x, cen, ctype*);

The function returns the standard deviation for a type 1 censored sample from a normal population, where *x* is the sample vector, *cen* is the censor vector with elements of unity for censored observations, and *ctype* is set to zero for left censored, and unity for right censored.

```
y = censtdc(x,cen,1);
```

COMBS *p* = COMBS (*v, k*);

Given an input of a scalar *n* or a vector *v* of length *n*, computes the matrix *p* with ${}^{n}C_{k}$ rows and *k* columns containing all possible ${}^{n}C_{k}$ combinations of the *k* elements.

```
p1 = COMBS(5,2);
let v = 1 4 5 7 9;
p2 = COMBS(v,2);
```

DECONV *x1* = DECONV (*z, x2*);

Computes the deconvolution of a vector.

```
z = CONV(x1,x2);
x2 = DECONV(z,x1);
x1 = DECONV(z,x2);
```

INTERP *yint* = INTERP (*y, x, xtarg*);

Computes the univariate interpolation *yint* for the target points *xtarg*, given a grid *x* with associated function values *y*.

```
let xint = .3 .4 .5;
yint = INTERP(y, x, xint);
```

INTERP2 *val* = INTERP2 (*y, x, A, ytarg, xtarg*);

Computes the two dimensional interpolation on the *m*x*n* table *A*, tabulated at the grid points defined by the *m*x1 vector*y* and the *n*x1 vector*x* at the target points *ytarg* and *xtarg*. This is a table lookup function.

```
let x = .3 .4 .5;
y = seqa(.1,.1,10);
let A[10,3] =  .....
val = INTERP2(y, x, A, .75, .35);
```

Returns the scalar interpolated value of A evaluated at {.75, .35}.

ISCHAR *y* = ISCHAR (*x*);

Returns y = 1 if the matrix *x* contains a character, else returns y = 0.

ISEMPTY *y* = ISEMPTY (*x*);

Returns y = 1 if *x* is an empty string or matrix, else returns y = 0.

MPRINT MPRINT (*x, rowname, colname, title, pause*);

Prints out a formatted matrix *x*, with row and column names. Output waits for a key input if *pause* is true. This is a GAUSSX command.

PERMS *p* = PERMS (*v*);

Given an input of a scalar *n* or a vector of length *n*, computes the matrix *p* with *n*! rows and *n* columns containing all possible permutations of the *n* elements.

```
p1 = PERMS(3);
let v = 1 4 5;
p2 = PERMS(v);
```

POLYDIV *psi* = POLYDIV (*theta*, *phi*, *n*);

Computes the polynomial division psi(B) = theta(B)/phi(B) where *theta* and *phi* are input polynomial coefficient vectors and *psi* is an (n+1 x 1) polynomial coefficient vector.

POLYINV *psi* = POLYINV (*phi*, *n*);

Computes the polynomial inversion psi(B) = 1/phi(B), where *psi* is an (n+1 x 1) polynomial coefficient vector.

SCALZERO *y* = SCALZERO (*x*);

Returns y = 1 if *x* is a scalar equal to zero, else returns y = 0.

WAITKEY WAITKEY (*pause*);

Prompts for a key input if *pause* is true. This is a GAUSSX command.

XGAMMA *y* = XGAMMA (*x*);

Computes gamma(x), where *x* takes all real values $-\infty < x < \infty$.

XPAND *xmat* = XPAND (*x*, *p*);

xmat consists of all own product and cross product terms of *x*, for all powers from zero (a constant) up to and including *p*, without repetitions. Thus if *x* in an *nxk* matrix, and $p = 2$, then *xmat* will consist of a matrix with $m = 1 + k + .5 * k * (k + 1)$ columns.

Example

```
library gaussx;
let phi = .7 .2;
r = acf(0,phi,0,10);
```

This example shows how the autocorrelation function can be derived for a second order AR process;

A.7 Trouble Shooting

A.7.1 Windows

1. GAUSS failed to compile all the GAUSSX procedures. This can be done manually. Enter GAUSS, and at the GAUSS prompt, type:

   ```
   run c:\gauss\gsx\gaussx.cpl; ←┘
   ```

2. Some of the GAUSSX statements described in the GAUSSX help file do not seem to work.

   The help file has been written to include all versions of GAUSSX, up to 8.1.1. The actual set of commands that can be used depend on the version of GAUSSX that you actually have installed.

3. On running a GAUSSX command file, a whole lot of errors, followed by a number of undefined symbols.

   Ensure that the Option\Parse is checked; the File Type on the top RHS of the GAUSSX Project Options window should display "Gaussx".

4. On running a GAUSS command file, a whole lot of errors, followed by a number of undefined symbols, or errors called by PARSE.

   Ensure that the Option\Parse is unchecked; the File Type on the top RHS of the GAUSSX Project Options window should display "Gauss".

A.7.2 UNIX

1. GAUSSX cannot be found with the command: run gaussx.

   Make sure that GAUSS is launched from its own directory, or use the full path name.

2. GAUSSX loads, but dies at the menu. All the paths on the file /gauss/gsx/gaussx.cfg must be complete and valid, and you must have write permission for the output file, and the work and scratch paths. For network situations, see Chapter 2.

A.8 Statlib Reference

Continuous Distributions

For x to be distributed by a continuous distribution, x must be continuous and smooth over the specified range.

Beta Distribution

PDF

$$\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$

CDF

$$\int_{-\infty}^{x}\frac{1}{B(\alpha,\beta)}t^{\alpha-1}(1-t)^{\beta-1}dt$$

where B is the Beta function.

Range $0 \leq x \leq 1$.
Shape1 parameter $\alpha > 0$.
Shape2 parameter $\beta > 0$.

Beta Distribution with lower and upper threshold

PDF

$$\frac{1}{B(\alpha,\beta)}\left(\frac{x-\theta_1}{\theta_2-\theta_1}\right)^{\alpha-1}\left(\frac{\theta_2-x}{\theta_2-\theta_1}\right)^{\beta-1}$$

CDF

$$\int_{-\infty}^{\frac{x-\theta_1}{\theta_2-\theta_1}}\frac{1}{B(\alpha,\beta)}t^{\alpha-1}(1-t)^{\beta-1}dt$$

where B is the Beta function.

Range $\theta 1 \leq x < \theta 2$.
Shape1 parameter $\alpha > 0$.
Shape2 parameter $\beta > 0$.
Lower threshold θ_1
Upper threshold θ_2

Notes: Estimation of the 4 parameter Beta distribution is undertaken in two parts. In the first part, initial parameter estimates are derived using the method of moments. The threshold parameters are then held using these values, and the shape parameters are then estimated using maximum likelihood.

BoxCox Distribution

PDF

$$\frac{x^{\lambda-1}}{\sigma\sqrt{2\pi}} e^{-((x^{\lambda}-1)/\lambda-\mu)^2/2\sigma^2}$$

CDF

$$\int_{-\infty}^{(x^{\lambda}-1)/\lambda} \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-\mu)^2/2\sigma^2} dt$$

Range $0 < x < \infty$.
Location parameter, μ, the mean.
Scale parameter, $\sigma > 0$, the standard deviation.
Shape parameter λ.

Notes: The concentrated likelihood is used in the ML estimation. This implies that the location and scale parameters are not estimated freely, but are derived as the mean and standard deviation of the BoxCox transformed variate.

BoxCox Distribution with threshold

PDF

$$\frac{(x-\theta)^{\lambda-1}}{\sigma\sqrt{2\pi}} e^{-(((x-\theta)^{\lambda}-1)/\lambda-\mu)^2/2\sigma^2}$$

CDF

$$\int_{-\infty}^{[(x-\theta)^{\lambda}-1]/\lambda} \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-\mu)^2/2\sigma^2} dt$$

Range $0 < x - \theta < \infty$.
Location parameter, μ, the mean.
Scale parameter, $\sigma > 0$, the standard deviation.
Shape parameter λ.
Threshold parameter $\theta < \min(x)$.

Notes: The concentrated likelihood is used in the ML estimation. This implies that the location and scale parameters are not estimated freely, but are derived as the mean and standard deviation of the BoxCox transformed variate.

Burr Distribution

PDF

$$\frac{ck(x/\beta)^{c-1}}{\beta(1+(x/\beta)^c)^{k+1}}$$

CDF

$$1-(1+(x/\beta)^c)^{-k}$$

Range $0 \le x < \infty$.
Scale parameter $\beta > 0$.
Shape parameter, $c > 0$.
Shape parameter, $k > 0$.

Burr Distribution with threshold

PDF

$$\frac{ck((x-\theta)/\beta)^{c-1}}{\beta(1+((x-\theta)/\beta)^c)^{k+1}}$$

CDF

$$1-(1+((x-\theta)/\beta)^c)^{-k}$$

Range $0 \leq x - \theta < \infty$.
Scale parameter $\beta > 0$.
Shape parameter, $c > 0$.
Shape parameter, $k > 0$.
Threshold parameter $\theta < \min(x)$.

Cauchy Distribution

PDF

$$\left(\pi\beta\left[1 + \left(\frac{x-\alpha}{\beta}\right)^2\right]\right)^{-1}$$

CDF

$$0.5 + \frac{1}{\pi}\tan^{-1}\left(\frac{x-\alpha}{\beta}\right)$$

Range $-\infty < x < \infty$.
Location parameter α, the median.
Scale parameter $\beta > 0$.

Chi-Squared Distribution

PDF

$$\frac{x^{v/2}\exp(-x/2)}{2^{v/2}\Gamma(v/2)}$$

CDF

$$\frac{\gamma(v/2, x/2)}{\Gamma(v/2)}$$

where $\Gamma(k)$ is the Gamma function, and $\gamma(k, z)$ is the lower incomplete Gamma function.

Range $0 \leq x \leq \infty$.
Shape parameter $v > 0$, the degrees of freedom.

Chi-Squared Distribution with threshold

PDF

$$\frac{(x-\theta)^{v/2}\exp(-(x-\theta)/2)}{2^{v/2}\Gamma(v/2)}$$

CDF

$$\frac{\gamma(v/2,(x-\theta)/2)}{\Gamma(v/2)}$$

where $\Gamma(k)$ is the Gamma function, and $\gamma(k,z)$ is the lower incomplete Gamma function.

Range $0 \le x-\theta \le \infty$.
Shape parameter $v > 0$, the degrees of freedom.
Threshold parameter $\theta < \min(x)$.

Chisq Distribution with scale

PDF

$$\frac{(x/\beta)^{.5v-1}e^{-0.5x/\beta}}{\beta\, 2^{.5v}\Gamma(.5v)}$$

CDF

$$\int_0^{.5x/\beta}\frac{e^{-t}t^{.5v-1}}{\Gamma(.5v)}dt$$

where Γ is the gamma function.

Range $0 \le x < \infty$.
Shape parameter $v > 0$.
Scale parameter $\beta > 0$.

Chisq Distribution with scale and threshold

PDF

$$\frac{((x-\theta)/\beta)^{.5\nu-1}e^{-0.5(x-\theta)/\beta}}{\beta\, 2^{.5\nu}\Gamma(.5\nu)}$$

CDF

$$\int_0^{.5(x-\theta)/\beta} \frac{e^{-t}t^{.5\nu-1}}{\Gamma(.5\nu)}dt$$

where Γ is the gamma function.

Range $0 \le x - \theta < \infty$.
Shape parameter $\nu > 0$.
Scale parameter $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Erf Distribution

PDF

$$\frac{\beta e^{-(\beta x)^2}}{\sqrt{\pi}}$$

CDF

$$\int_{-\infty}^{x} \frac{e^{-(\beta t)^2}}{\sqrt{\pi}}dt$$

Range $-\infty < x < \infty$.
Scale parameter $\beta > 0$.

Exponential Distribution

PDF

$$\frac{e^{-x/\alpha}}{\alpha}$$

CDF

$$1 - e^{-x/\alpha}$$

Range $0 \leq x < \infty$.
Scale parameter, $\alpha > 0$, the mean.

Exponential Distribution with threshold

PDF

$$\frac{e^{-(x-\theta)/\alpha}}{\alpha}$$

CDF

$$1 - e^{-(x-\theta)/\alpha}$$

Range $0 \leq x - \theta < \infty$.
Scale parameter, $\alpha > 0$, the mean.
Threshold parameter $\theta < \min(x)$.

F Distribution

PDF

$$\frac{(v/w)^{v/2} x^{(v/2-1)}}{B(v/2, w/2)[1 + (v/w)x]^{(v+w)/2)}}$$

CDF

$$I_{\frac{vx}{vx+w}}(v/2, w/2)$$

where $B(a, b)$ is the Beta function, and where $I_x(a, b)$ is the regularized incomplete Beta function.

Range $0 \leq x \leq \infty$.
Shape1 parameter $v > 0$, integer, first degrees of freedom.
Shape2 parameter $w > 0$, integer, second degrees of freedom.

F Distribution with threshold

PDF

$$\frac{(v/w)^{v/2}(x-\theta)^{(v/2-1)}}{B(v/2,w/2)[1+(v/w)(x-\theta)]^{(v+w)/2)}}$$

CDF

$$I_{\frac{vx}{v(x-\theta)+w}}(v/2,w/2)$$

where B is the Beta function, and where $I_x(a,b)$ is the regularized incomplete Beta function.

Range $0 \leq x-\theta < \infty$.
Shape1 parameter $v > 0$, integer, first degrees of freedom.
Shape2 parameter $w > 0$, integer, second degrees of freedom.
Threshold parameter $\theta < \min(x)$.

F Distribution with scale

PDF

$$\frac{1}{x\,\mathrm{B}(.5\nu,.5\omega)}\sqrt{\frac{(\nu\,x/\alpha)^{\nu}\;\omega^{\omega}}{(\nu\,x/\alpha+\omega)^{\nu+\omega}}}$$

CDF

$$\int_{-\infty}^{z}\frac{1}{B(.5\,\nu,.5\,\omega)}t^{.5\nu-1}(1-t)^{.5\omega-1}dt$$

where

$$z=\frac{(\nu\,x)}{(\nu\,x+\alpha\omega)}$$

and where B is the Beta function.

Range $0 \leq x < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\nu > 0$.
Shape parameter $\omega > 0$.

F Distribution with scale and threshold

PDF

$$\frac{1}{(x-\theta)\,\mathrm{B}(.5\nu,.5\omega)}\sqrt{\frac{(\nu\,(x-\theta)/\alpha)^{\nu}\;\omega^{\omega}}{(\nu\,(x-\theta)/\alpha+\omega)^{\nu+\omega}}}$$

CDF

$$\int_{-\infty}^{z}\frac{1}{B(.5\,\nu,.5\,\omega)}t^{.5\nu-1}(1-t)^{.5\omega-1}dt$$

where

$$z=\frac{(\nu\,(x-\theta))}{(\nu\,x+\alpha\omega)}$$

and where B is the Beta function.

Range $0 \le x-\theta < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\nu > 0$.
Shape parameter $\omega > 0$. Threshold parameter $\theta < \min(x)$.

Fatigue Life Distribution

PDF

$$\frac{\sqrt{\frac{x}{\beta}}+\sqrt{\frac{\beta}{x}}}{2\gamma x)}\phi\left(\frac{\sqrt{\frac{x}{\beta}}-\sqrt{\frac{\beta}{x}}}{\gamma}\right)$$

CDF

$$\Phi\left(\frac{\sqrt{x}-\sqrt{\frac{1}{x}}}{\gamma}\right)$$

where $\phi(x)$ and $\Phi(x)$ are respectively the PDF and CDF of the standard normal distribution.

Range $0 < x < \infty$.
Scale parameter, $\beta > 0$.
Shape parameter $\gamma > 0$.

Notes: This is also known as the Birnbaum Saunders distribution.

Fatigue Life Distribution with threshold

PDF

$$\frac{\sqrt{\frac{x-\theta}{\beta}} + \sqrt{\frac{\beta}{x-\theta}}}{2\gamma\,(x-\theta)}\phi\left(\frac{\sqrt{\frac{x-\theta}{\beta}} - \sqrt{\frac{\beta}{x-\theta}}}{\gamma}\right)$$

CDF

$$\Phi\left(\frac{\sqrt{x-\theta} - \sqrt{\frac{1}{x-\theta}}}{\gamma}\right)$$

where $\phi(x)$ and $\Phi(x)$ are respectively the PDF and CDF of the standard normal distribution.

Range $0 < x - \theta < \infty$.
Scale parameter, $\beta > 0$.
Shape parameter $\gamma > 0$.
Threshold parameter $\theta < \min(x)$.

Notes: This is also known as the Birnbaum Saunders distribution.

Fisk Distribution

PDF

$$\frac{(\beta/\alpha)(x/\alpha)^{\beta-1}}{\left[1 + (x/\alpha)^{\beta}\right]^2}$$

CDF

$$\frac{1}{1 + (x/\alpha)^{-\beta}}$$

Range $0 < x < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter, $\beta > 0$.

Fisk Distribution with threshold

PDF

$$\frac{(\beta/\alpha)((x-\theta)/\alpha)^{\beta-1}}{[1+((x-\theta)/\alpha)^{\beta}]^2}$$

CDF

$$\frac{1}{1+((x-\theta)/\alpha)^{-\beta}}$$

Range $0 < x - \theta < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter, $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Folded Normal Distribution

PDF

$$\frac{\sqrt{2\pi}}{\sigma}\cosh(\mu/\sigma^2)e^{-(x^2+\mu^2)/2\sigma^2}$$

CDF

$$\Phi\left(\frac{x-\mu}{\sigma}\right)-\Phi\left(\frac{-x-\mu}{\sigma}\right)$$

where $\Phi(x)$ is the CDF of the standard normal distribution.

Range $0 \le x < \infty$.
Location parameter, μ, the mean.
Scale parameter, $\sigma > 0$, the standard deviation.

Frechet Distribution

PDF

$$(\beta/\alpha)(\alpha/x)^{1+\beta}e^{-(\alpha/x)^{\beta}}$$

CDF

$$e^{-(\alpha/x)^{\beta}}$$

Range $0 \leq x < \infty$.
Scale parameter, $\alpha > 0$.
Shape parameter $\beta > 0$.

Frechet Distribution with threshold

PDF

$$(\beta/\alpha)(\alpha/(x-\theta))^{1+\beta}e^{-(\alpha/(x-\theta))^{\beta}}$$

CDF

$$e^{-(\alpha/(x-\theta))^{\beta}}$$

Range $0 \leq x - \theta < \infty$.
Scale parameter, $\alpha > 0$.
Shape parameter $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Gamma Distribution

PDF

$$\frac{(x/\alpha)^{\beta-1}e^{-x/\alpha}}{\alpha\Gamma(\beta)}$$

CDF

$$\int_0^{x/\alpha} \frac{e^{-t}t^{\beta-1}}{\Gamma(\beta)}dt$$

where $\Gamma(\beta)$ is the the Gamma function.

Range $0 \leq x < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.

Gamma Distribution with threshold

PDF

$$\frac{[(x-\theta)/\alpha])^{\beta-1} e^{-(x-\theta)/\alpha}}{\alpha\Gamma(\beta)}$$

CDF

$$\int_0^{(x-\theta)/\alpha} \frac{e^{-t} t^{\beta-1}}{\Gamma(\beta)} dt$$

where $\Gamma(\beta)$ is the the Gamma function:
Range $0 \leq x - \theta < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Generalized Error Distribution

PDF

$$\frac{\beta}{2\alpha\Gamma(1/\beta)} e^{-(|x-\mu|/\alpha)^{\beta}}$$

CDF

$$\frac{1}{2} + \text{sign}(x-\mu)\frac{\gamma\left(1/\beta, \left(\frac{|x-\mu|}{\alpha}\right)^{\beta}\right)}{2\Gamma(1/\beta)}$$

where $\Gamma(\beta)$ is the the Gamma function and $\gamma(k, z)$ is the lower incomplete Gamma function.

Range $-\infty \leq x < \infty$.
Location parameter μ. Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.

Notes: This is also known as the Exponential Power distribution or the Generalized Normal distribution.

Generalized Gamma Distribution

PDF

$$\frac{px^{(pk-1)}e^{(-x/\alpha^p)}}{\alpha^{kp}\Gamma(k)}$$

CDF

$$\gamma(k,(x/\alpha)^p)$$

where $\Gamma(\beta)$ is the the Gamma function and $\gamma(k,z)$ is the lower incomplete Gamma function.

Range $0 \leq x < \infty$.
Scale parameter $\alpha > 0$.
Shape1 parameter $k > 0$.
Shape1 parameter $p > 0$.

Generalized Gamma Distribution with threshold

PDF

$$\frac{p(x-\theta)^{(pk-1)}e^{(-(x-\theta)/\alpha^p)}}{\alpha^{kp}\Gamma(k)}$$

CDF

$$\gamma(k,[(x-\theta)/\alpha]^p)$$

where $\Gamma(\beta)$ is the the Gamma function and $\gamma(k,z)$ is the lower incomplete Gamma function.

Range $0 \leq x - \theta < \infty$.
Scale parameter $\alpha > 0$.
Shape1 parameter $k > 0$.
Shape1 parameter $p > 0$.
Threshold parameter $\theta < \min(x)$.

Generalized Logistic Distribution

PDF

$$\frac{\alpha e^{-(x-\mu)/\sigma}}{\sigma(1 + e^{-(x-\mu)/\sigma})^{1+\alpha}}$$

CDF

$$\frac{1}{(1 + e^{-(x-\mu)/\sigma})^{\alpha}}$$

Range $-\infty < x < \infty$.
Location parameter, μ.
Scale parameter $\sigma > 0$.
Skew parameter α, < 1 for left skew, > 1 for right skew.

Notes: This is a Type I Generalized Logistic distribution; it is also known as the Skew-Logistic distribution.

Generalized Pareto Distribution

PDF

$$\frac{1}{\alpha}\left(1 + \beta\,\frac{x-\mu}{\alpha}\right)^{-(1+1/\beta)}$$

CDF

$$1 - \left(1 + \beta\,\frac{x-\mu}{\alpha}\right)^{-1/\beta}$$

Range $0 < x < \infty$.
Location parameter μ.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.

Half Normal Distribution

PDF

$$\frac{\sqrt{2/\pi}}{\sigma^2} e^{-x^2/2\sigma^2}$$

CDF

$$\int_{-\infty}^{x} \frac{\sqrt{2/\pi}}{\sigma} e^{-t^2/2\sigma^2} dt - 1$$

Range $0 \leq x < \infty$.
Scale parameter, $\sigma > 0$, the standard deviation.

Half Normal Distribution with threshold

PDF

$$\frac{\sqrt{2/\pi}}{\sigma^2} e^{-(x-\theta)^2/2\sigma^2}$$

CDF

$$\int_{-\infty}^{x-\theta} \frac{\sqrt{2/\pi}}{\sigma} e^{-t^2/2\sigma^2} dt - 1$$

Range $0 \leq x - \theta < \infty$.
Scale parameter, $\sigma > 0$, the standard deviation.
Threshold parameter $\theta < \min(x)$.

Inverse Gamma Distribution

PDF

$$\frac{\alpha^\beta}{\Gamma(\beta)} x^{-\beta-1} e^{-\alpha/x}$$

CDF

$$\frac{\gamma(\beta, \alpha/x)}{\Gamma(\beta)}$$

where $\Gamma(s)$ is the the gamma function, and $\gamma(s, x)$ is the lower incomplete gamma function.

Range $0 \leq x < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.

Inverse Gamma Distribution with threshold

PDF

$$\frac{\alpha^{\beta}}{\Gamma(\beta)}(x-\theta)^{-\beta-1}e^{-\alpha/(x-\theta)}$$

CDF

$$\frac{\gamma(\beta, \alpha/(x-\theta))}{\Gamma(\beta)}$$

where $\Gamma(s)$ is the the gamma function, and $\gamma(s, x)$ is the incomplete gamma function.

Range $0 \leq x - \theta < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Inverse Gaussian Distribution

PDF

$$\left[\frac{\lambda}{2\pi x^3}\right]^{1/2} e^{-\lambda(x-\mu)^2/(2\mu^2 x)}$$

CDF

$$\Phi\left(\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu}-1\right)\right) + e^{2\lambda/\mu}\,\Phi\left(-\sqrt{\frac{\lambda}{x}}\left(\frac{x}{\mu}+1\right)\right)$$

where $\Phi(x)$ is the CDF of the standard normal distribution.

Range $0 < x < \infty$.
Location parameter, μ, the mean.
Shape parameter $\lambda > 0$.

Johnson_SB Distribution

PDF

$$\frac{\delta e^{-.5(\gamma+\delta \ln(z/(1-z)))^2}}{\lambda \sqrt{2\pi} z(1-z)}$$

CDF

$$\Phi\left[\gamma + \delta \ln\left(\frac{z}{1-z}\right)\right]$$

where $z = (x - \eta)/\lambda$ and $\Phi(x)$ is the CDF of the standard normal distribution.

Range $\eta < x < \eta + \lambda$.
Location parameter, η, the mean.
Scale parameter $\lambda > 0$.
Shape1 parameter γ
Shape2 parameter $\delta > 0$

Johnson_SL Distribution

PDF

$$\frac{\delta \phi\left[\gamma + \delta \ln\left(\frac{x-\eta}{\lambda}\right)\right]}{(x-\eta)}$$

CDF

$$\Phi\left[\gamma + \delta \ln\left(\frac{x-\eta}{\lambda}\right)\right]$$

where $\phi(x)$ and $\Phi(x)$ are respectively the PDF and CDF of the standard normal distribution.

Range $\eta < x < \infty$.

Location parameter, η, the mean.
Scale parameter $\lambda = 1$.
Shape1 parameter γ
Shape2 parameter $\delta > 0$

Johnson_SU Distribution

PDF

$$\frac{\delta e^{-.5(\gamma+\delta \sinh^{-1}(z))^2}}{\lambda\sqrt{2\pi(z^2+1)}}$$

CDF

$$\Phi(\gamma + \delta \sinh^{-1}(z))$$

where $z = (x - \eta)/\lambda$ and $\Phi(x)$ is the CDF of the standard normal distribution.

Range $-\infty < x < \infty$.
Location parameter, η, the mean.
Scale parameter $\lambda > 0$.
Shape1 parameter γ
Shape2 parameter $\delta > 0$

Laplace Distribution

PDF

$$\frac{1}{2\sigma}e^{-\frac{|x-\mu|}{\sigma}}$$

CDF

$$\frac{1}{2\sigma}e^{-\frac{\mu-x}{\sigma}} \quad \text{if } x < \mu$$

$$\frac{1}{2\sigma}e^{-\frac{x-\mu}{\sigma}} \quad \text{if } x \geq \mu$$

Range $-\infty < x < \infty$.

Location parameter, μ, the mean.
Scale parameter $\sigma > 0$.

Largest Extreme Value Distribution

PDF

$$\frac{1}{\sigma} e^{-(x-\mu)/\sigma} e^{-e^{-(x-\mu)/\sigma}}$$

CDF

$$e^{-e^{-(x-\mu)/\sigma}}$$

Range $-\infty < x < \infty$.
Location parameter, μ, the mode.
Scale parameter $\sigma > 0$.

Notes: The Gumbel distribution is equivalent to the Largest Extreme Value.

Levy Distribution

PDF

$$\sqrt{\frac{\sigma}{2\pi}} \frac{e^{-\sigma/2x}}{x^{3/2}}$$

CDF

$$\text{erfc}\left(\sqrt{\sigma/2x}\right)$$

Range $0 < x < \infty$.
Scale parameter $\sigma > 0$.

LogGamma Distribution

PDF

$$\frac{\ln(x)^{\beta-1} e^{-\ln(x)/\alpha}}{x\alpha^{\beta}\Gamma(\beta)}$$

CDF

$$\int_0^{\ln(x)/\alpha} \frac{e^{-t}t^{\beta-1}}{\Gamma(\beta)}dt$$

where Γ is the gamma function.

Range $0 \le x < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.

LogGamma Distribution with threshold

PDF

$$\frac{\ln(x-\theta)^{\beta-1}e^{-\ln(x-\theta)/\alpha}}{(x-\theta)\alpha^{\beta}\Gamma(\beta)}$$

CDF

$$\int_0^{\ln(x-\theta)/\alpha} \frac{e^{-t}t^{\beta-1}}{\Gamma(\beta)}dt$$

where Γ is the gamma function.

Range $0 < x - \theta < \infty$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Logistic Distribution

PDF

$$\frac{e^{(x-\mu)/\sigma}}{\sigma(1+e^{(x-\mu)/\sigma})^2}$$

CDF

$$\frac{1}{1+e^{-(x-\mu)/\sigma}}$$

Range $-\infty < x < \infty$.
Location parameter, μ, the mean.
Scale parameter $\sigma > 0$.

Loglogistic Distribution

PDF

$$\frac{e^{(\ln(x)-\mu)/\sigma}}{x\sigma(1+e^{(\ln(x)-\mu)/\sigma})^2}$$

CDF

$$\frac{1}{1+e^{-(\ln(x)-\mu)/\sigma}}$$

Range $0 < x < \infty$.
Location parameter, μ, the mean.
Scale parameter $\sigma > 0$.

Loglogistic Distribution with threshold

PDF

$$\frac{e^{(\ln(x-\theta)-\mu)/\sigma}}{(x-\theta)\sigma(1+e^{(\ln(x-\theta)-\mu)/\sigma})^2}$$

CDF

$$\frac{1}{1+e^{-(\ln(x-\theta)-\mu)/\sigma}}$$

Range $0 < x - \theta < \infty$.
Location parameter, μ, the mean.
Scale parameter $\sigma > 0$.
Threshold parameter $\theta < \min(x)$.

LogNormal Distribution

PDF

$$\frac{1}{x\sqrt{2\pi\sigma^2}}e^{-.5(\ln(x)-\mu)^2/\sigma^2}$$

CDF

$$\int_{-\infty}^{x} \frac{1}{t\sigma\sqrt{2\pi}} e^{-.5(\ln(t)-\mu)^2/\sigma^2} dt$$

Range $0 < x < \infty$.
Scale parameter, μ, the mean of $\ln(x)$.
Shape parameter, $\sigma > 0$, the standard deviation of $\ln(x)$.

LogNormal Distribution with threshold

PDF

$$\frac{1}{(x-\theta)\sqrt{2\pi\sigma^2}} e^{-.5(\ln(x-\theta))-\mu)^2/\sigma^2}$$

CDF

$$\int_{-\infty}^{x-\theta} \frac{1}{t\sigma\sqrt{2\pi}} e^{-.5(\ln(t)-\mu)^2/\sigma^2} dt$$

Range $0 < x - \theta < \infty$.
Scale parameter, μ, the mean of $\ln(x)$.
Shape parameter, $\sigma > 0$, the standard deviation of $\ln(x)$.
Threshold parameter $\theta < \min(x)$.

Maxwell Boltzmann Distribution

PDF

$$\sqrt{\frac{2}{\pi}} \frac{x^2 e^{-x^2/(2a^2)}}{a^3}$$

CDF

$$\gamma\left(1.5, \frac{x^2}{2a^2}\right)$$

where $\gamma(s, z)$ is the lower incomplete Gamma function.

Range $0 < x < \infty$.
Scale parameter $a > 0$.

Non-Central Chi-Squared Distribution

PDF

$$.5e^{-(x+.5\lambda)}\left(\frac{x}{\lambda}\right)^{.25\nu-.5} I_{.5\nu-1}(\sqrt{\lambda x})$$

CDF

$$\sum_{j=0}^{\infty} e^{-.5\lambda}\frac{(.5\lambda)^j}{j!}\frac{\gamma(j+.5\nu, .5x)}{\Gamma(j+.5k)}$$

where $\Gamma(s)$ is the gamma function, $\gamma(s, x)$ is the lower incomplete gamma function. and I is the modified Bessel function of the first kind.

Range $0 \leq x < \infty$.
Shape parameter $\nu > 0$.
Non-centrality parameter $\lambda > 0$.

Non-Central Chi-Squared Distribution with threshold

PDF

$$.5e^{-(x-\theta+.5\lambda)}\left(\frac{x-\theta}{\lambda}\right)^{.25\nu-.5} I_{.5\nu-1}(\sqrt{\lambda(x-\theta)})$$

CDF

$$\sum_{j=0}^{\infty} e^{-.5\lambda}\frac{(.5\lambda)^j}{j!}\frac{\gamma(j+.5\nu, .5(x-\theta))}{\Gamma(j+.5k)}$$

where $\Gamma(s)$ is the gamma function, $\gamma(s, x)$ is the lower incomplete gamma function. and I is the modified Bessel function of the first kind.

Range $0 \leq x - \theta < \infty$.
Shape parameter $\nu > 0$.

Non-centrality parameter $\lambda > 0$.
Threshold parameter $\theta < \min(x)$.

Non-Central F Distribution

PDF

$$\sum_{k=0}^{\infty} \frac{e^{-.5\lambda}(.5\lambda)^k}{B(.5\nu_2,\ .5\nu_1 + k)\,k!} \left(\frac{\nu_1}{\nu_2}\right)^{.5\nu_1+k} \left(\frac{\nu_2}{\nu_2 + \nu_1 x}\right)^{.5(\nu_1+\nu_2)+k} x^{.5\nu_1-1+k}$$

CDF

$$\sum_{j=0}^{\infty} \frac{(.5\lambda)^j\ e^{-.5\lambda}}{j!} B(z;\ .5\nu_1 + j,\ .5\nu_2)$$

where

$$z = \frac{\nu_1\, x}{\nu_1 x + \nu_2}$$

where $B(a, b)$ is the beta function, and $B(z;\ a, b)$ is the incomplete beta function.

Range $0 \le x < \infty$.
Shape parameter $\nu_1 > 0$.
Shape parameter $\nu_2 > 0$.
Non-centrality parameter $\lambda > 0$.

Non-Central F Distribution with threshold

PDF

$$\sum_{k=0}^{\infty} \frac{e^{-.5\lambda}(.5\lambda)^k}{B(.5\nu_2,\ .5\nu_1 + k)\,k!} \left(\frac{\nu_1}{\nu_2}\right)^{.5\nu_1+k} \left(\frac{\nu_2}{\nu_2 + \nu_1(x-\theta)}\right)^{.5(\nu_1+\nu_2)+k} (x-\theta)^{.5\nu_1-1+k}$$

CDF

$$\sum_{j=0}^{\infty} \frac{(.5\lambda)^j\ e^{-.5\lambda}}{j!} B(z;\ .5\nu_1 + j,\ .5\nu_2)$$

where

$$z = \frac{\nu_1(x-\theta)}{\nu_1(x-\theta)+\nu_2}$$

where $B(a,b)$ is the beta function, and $B(z;\, a,b)$ is the incomplete beta function.

Range $0 \le x-\theta < \infty$.
Shape parameter $\nu_1 > 0$.
Shape parameter $\nu_2 > 0$.
Non-centrality parameter $\lambda > 0$. Threshold parameter $\theta < \min(x)$.

Non-Central T Distribution

PDF

$$\frac{\nu^{.5\nu} e^{-\nu\lambda^2/(2x^2+2\nu)}}{\sqrt{\pi}\Gamma(.5\nu)\, 2^{.5(\nu-1)}(x^2+\nu)^{.5(\nu+1)}} \int_0^\infty t^\nu e^{-.5(t-\lambda x/\sqrt{x^2+\nu})^2}\, dt$$

CDF ($x \ge 0$)

$$\Phi(-\lambda) + \frac{1}{2}\sum_{j=0}^{\infty}\left[p_j I_z\left(j+\frac{1}{2},\frac{\nu}{2}\right) + q_j \beta_z\left(j+1,\frac{\nu}{2}\right)\right]$$

where

$$\begin{aligned} z &= \frac{x^2}{x^2+\nu} \\ p_j &= \frac{e^{-.5\lambda^2}}{j!}\left(\frac{\lambda^2}{2}\right)^j \\ q_j &= \frac{\lambda e^{-.5\lambda^2}}{\sqrt{2}\Gamma(j+3/2)}\left(\frac{\lambda^2}{2}\right)^j \end{aligned}$$

and where Φ is the standard normal CDF, Γ is the gamma function, and $I_z(a,b)$ is the regularized incomplete beta function.

Range $-\infty \leq x < \infty$.
Shape parameter $\nu > 0$.
Non-centrality parameter $\lambda > 0$.

Normal Distribution

PDF

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-.5(x-\mu)^2/\sigma^2}$$

CDF

$$\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-.5(t-\mu)^2/\sigma^2} dt$$

Range $-\infty < x < \infty$.
Location parameter, μ, the mean.
Scale parameter, $\sigma > 0$, the standard deviation.

Pareto Distribution

PDF

$$\frac{\alpha\, x_{\mathrm{m}}^{\alpha}}{x^{\alpha+1}}$$

CDF

$$1 - \left(\frac{x_{\mathrm{m}}}{x}\right)^{\alpha}$$

Range $x_m < x < \infty$.
Location parameter $x_m > 0$, the minimum of x.
Shape parameter $\alpha > 0$

Pearson III Distribution

PDF

$$\frac{[(x-\mu)/\alpha])^{\beta-1} e^{-(x-\mu)/\alpha}}{\alpha\Gamma(\beta)}$$

CDF

$$\int_0^{(x-\mu)/\alpha} \frac{e^{-t}t^{\beta-1}}{\Gamma(\beta)}dt$$

where $\Gamma(\beta)$ is the the Gamma function:
Range $0 \leq x - \mu < \infty$.
Location parameter $\mu < \min(x)$.
Scale parameter $\alpha > 0$.
Shape parameter $\beta > 0$.

PERT Distribution

PDF

$$\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$

CDF

$$\int_{-\infty}^{x} \frac{1}{B(\alpha,\beta)}t^{\alpha-1}(1-t)^{\beta-1}dt$$

where B is the Beta function.

Range $lb \leq z \leq ub$.
Parameter lb : $lb < z_{min}$, the lower bound of z.
Parameter ub : $z_{max} < ub$, the upper bound of z.
Parameter η : $lb < \eta < ub$, the mode of z.

Notes: The PERT argument z and the three parameters, lb, ub and η are transformed using the PERT transform; the resulting argument, x is distributed Beta, with shape parameters α and β.

Power Distribution

PDF

$$\frac{\nu(x-a)^{\nu-1}}{(b-a)^{\nu}}$$

CDF

$$\frac{(x-a)^{\nu}}{(b-a)^{\nu}}$$

Range $a \leq x \leq b$.
Lowerbound parameter $a > 0$.
Upperbound parameter b.
Shape parameter $\nu > 0$.

Rayleigh Distribution

PDF

$$\frac{x}{\alpha^2} e^{-x^2/(2\alpha^2)}$$

CDF

$$1 - e^{-x^2/(2\alpha^2)}$$

Range $0 \leq x < \infty$.
Scale parameter, $\alpha > 0$.

Reciprocal Distribution

PDF

$$\frac{1}{x(\ln(b) - \ln(a))}$$

CDF

$$\frac{\ln(x) - \ln(a)}{\ln(b) - \ln(a)}$$

Range $a \leq x \leq b$.
Lowerbound parameter $a > 0$.
Upperbound parameter b.

Skew Normal Distribution

PDF

$$\frac{2}{\sigma}\Phi\left(\frac{\alpha(x-\mu)}{\sigma}\right)\phi\left(\frac{x-\mu}{\sigma}\right)$$

CDF

$$2\,\text{cdfbvn}\left(\frac{x-\mu}{\sigma}, 0, \frac{-\alpha}{\sqrt{1+\alpha^2}}\right)$$

where $\phi(x)$ and $\Phi(x)$ are respectively the PDF and CDF of the standard normal distribution, and cdfbvn is the cumulative standardized bivariate normal distribution.

Range $-\infty < x < \infty$.
Location parameter, μ.
Scale parameter $\sigma > 0$.
Skew parameter α, negative for left skew, positive for right skew.

Smallest Extreme Value Distribution

PDF

$$\frac{1}{\sigma}e^{(x-\mu)/\sigma}e^{-e^{(x-\mu)/\sigma}}$$

CDF

$$1 - e^{-e^{(x-\mu)/\sigma}}$$

Range $-\infty < x < \infty$.
Location parameter, μ, the mode.
Scale parameter $\sigma > 0$.

Student's T Distribution

PDF

$$\frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1+\frac{x^2}{\nu}\right)^{-(\frac{\nu+1}{2})}$$

CDF

$$\frac{1}{2} + x\Gamma\left(\frac{\nu+1}{2}\right)\frac{{}_2F_1\left(\frac{1}{2}, \frac{\nu+1}{2}; \frac{3}{2}; -\frac{x^2}{\nu}\right)}{\sqrt{\pi\nu}\,\Gamma(\frac{\nu}{2})}$$

where ${}_2F_1$ is the hypergeometric function.

Range $-\infty < x < \infty$.
Shape parameter $\nu > 0$, degrees of freedom.

Student's T Distribution with location and scale

PDF

$$\frac{1}{\alpha\ \sqrt{\pi\,\nu}}\frac{\Gamma(.5(\nu+1))}{\Gamma(.5\nu)}\left(\frac{\nu\alpha^2}{(x-\mu)^2+\nu\alpha^2}\right)^{.5(\nu+1)}$$

CDF

$$\begin{aligned} &.5 + .5\,I_z(.5, .5\nu) \quad && x \geq 0 \\ &.5 - .5\,I_z(.5, .5\nu) \quad && x < 0 \end{aligned}$$

where

$$z \;=\; \frac{(x-\mu)^2}{(x-\mu)^2+\nu\alpha^2}$$

and where Γ is the gamma function, and $I_z(a, b)$ is the regularized incomplete beta function.

Range $-\infty \leq x < \infty$.
Location parameter μ.
Scale parameter $\alpha > 0$.
Shape parameter $\nu > 0$.

Triangular Distribution

PDF

$$\frac{2(x-a)}{(b-a)(c-a)} \quad \text{for } a \le x \le c$$

$$\frac{2(b-x)}{(b-a)(b-c)} \quad \text{for } c \le x \le b$$

CDF

$$\frac{(x-a)^2}{(b-a)(c-a)} \quad \text{for } a \le x \le c$$

$$1 - \frac{(b-x)^2}{(b-a)(b-c)} \quad \text{for } c \le x \le b$$

Range $a \le x \le b$.
Parameter a : $a \le x_{min}$, the lower bound of x.
Parameter b : $x_{max} \le b$, the upper bound of x.
Parameter c : $a < c < b$, the mode of x.

Uniform Distribution

PDF

$$\frac{1}{b-a}$$

CDF

$$\frac{x-a}{b-a}$$

Range $a \le x \le b$.
Parameter a : $a \le x_{min}$, the lower bound of x.
Parameter b : $x_{max} \le b$, the upper bound of x.

Von Mises Distribution

PDF

$$\frac{e^{\kappa \cos(x-\mu)}}{2\pi I_0(\kappa)}$$

CDF

$$\frac{1}{2\pi}\left(x+\frac{2}{I_0(\kappa)}\sum_{j=1}^{\infty} I_j(\kappa)\frac{\sin[j(x-\mu)]}{j}\right)$$

where $I_j(x)$ is the modified Bessel function of order j.

Range $0 \le x < 2\pi$.
Location parameter, μ : $0 \le \mu \le 2\pi$.
Shape parameter $\kappa > 0$.

Weibull Distribution

PDF

$$\frac{\beta x^{\beta-1}}{\alpha^{\beta}} e^{-(x/\alpha)^{\beta}}$$

CDF

$$1 - e^{-(x/\alpha)^{\beta}}$$

Range $0 \le x < \infty$.
Scale parameter, $\alpha > 0$, the characteristic life.
Shape parameter $\beta > 0$.

Weibull Distribution with threshold

PDF

$$\frac{\beta x^{\beta-1}}{\alpha^{\beta}} e^{-[(x-\theta)/\alpha]^{\beta}}$$

CDF

$$1 - e^{-[x-\theta)/\alpha]^{\beta}}$$

Range $0 \leq x - \theta < \infty$.
Scale parameter, $\alpha > 0$.
Shape parameter $\beta > 0$.
Threshold parameter $\theta < \min(x)$.

Discrete Distributions

For x to be distributed by a discrete distribution, x must only take discrete values over the specified range. With the exception of the Step distribution, x must be integer.

Bernoulli Distribution

The Bernoulli distribution takes a value 1 with probability p and value 0 with probability $1 - p$.

PDF

$$xp + (1 - x)(1 - p)$$

CDF

$$1 - p + px$$

Support $x \in \{0, 1\}$
Probability parameter, p : $0 \leq p \leq 1$.

Binomial Distribution

The binomial pdf is the probability of x successes in n independent trials, where p is the probability of success in any given trial.

PDF

$$\left(\begin{array}{c} n \\ x \end{array} \right) p^x (1 - p)^{n-x}$$

CDF

$$I_{1-p}(n - x, x + 1)$$

where $I_x(a, b)$ is the regularized incomplete Beta function.

Support $x \in \{0, \ldots, n\}$
Probability parameter, p : $0 \leq p \leq 1$.
Trials parameter, n : $n > 0$.

Geometric Distribution

The geometric pdf is the probability of x failures before a success, where p is the probability of success in any given trial.

PDF

$$p(1-p)^x$$

CDF

$$1-(1-p)^{x+1}$$

Support $x \in \{1, 2, 3, \ldots\}$
Probability parameter, p : $0 \leq p \leq 1$.

Hypergeometric Distribution

The Hypergeometric pdf is the probability of drawing x successes in n draws, without replacement, from a population of size N which contains m successes.

PDF

$$\frac{\binom{m}{x}\binom{N-m}{n-x}}{\binom{N}{n}}$$

CDF

$$\sum_{i=0}^{x}\left(\frac{\binom{m}{i}\binom{N-m}{n-i}}{\binom{N}{n}}\right)$$

Support $x \in \{1, 2, 3, \ldots\}$
Population parameter, N : $N > 0$.
Success parameter, m : $m > 0$.
Sample parameter, n : $n > 0$.

Logarithmic Distribution

The Logarithmic pdf is a one parameter generalized power series distribution.

PDF

$$\frac{-p^x}{x \ln(1-p)}$$

CDF

$$1 + \frac{B(p; x+1, 0)}{\ln(1-p)}$$

where B(x; a,b) is the incomplete beta function.

Support $x \in \{1, 2, 3, \ldots\}$
Probability parameter, p : $0 \leq p \leq 1$.

Negative Binomial Distribution

The Negative Binomial pdf is the probability of achieving r failures before the xth success, with p being the probability of a success.

PDF

$$\binom{x+r-1}{r-1}(1-p)^r p^x$$

CDF

$$1 - I_p(x+1,\ r)$$

where $I_x(a, b)$ is the regularized incomplete Beta function.

Support $x \in \{1, 2, 3, \ldots\}$
Probability parameter, p : $0 \leq p \leq 1$.
Failure parameter, r : $r > 0$.

Poisson Distribution

The Poisson pdf is the probability x events occurring within a period, where λ is the expected number of events in that period.

PDF

$$\frac{\lambda^x}{x!} e^{-\lambda}$$

CDF

$$e^{-\lambda} \sum_{i=0}^{x} \left(\frac{\lambda^i}{i!} \right)$$

Support $x \in \{1, 2, 3, \ldots\}$
Event parameter, λ : $r > 0$, the mean of x

Step Distribution

The step pdf is the same for each step.

PDF

$$\frac{s}{b - a + s}$$

CDF

$$\frac{x - a + s}{b - a + s}$$

Support $x \in \{a, \ldots, b\}$
Parameter a : $a \leq x_{min}$, the lower bound of x.
Parameter b : $x_{max} \leq b$, the upper bound of x.
Parameter s, the stepsize.

Uniform Distribution

The Uniform pdf is the same for each outcome.

PDF

$$\frac{1}{b-a+1}$$

CDF

$$\frac{x-a+1}{b-a+1}$$

Support $x \in \{a, \ldots, b\}$
Parameter a : $a \leq x_{min}$, the lower bound of x.
Parameter b : $x_{max} \leq b$, the upper bound of x.

Beta Function

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

Gamma Function

$$\Gamma(\beta) = \int_0^\infty e^{-t}t^{\beta-1}dt \qquad (\beta > 0)$$

Incomplete Beta Function

$$B(x;\, a, b) = \int_0^x t^{a-1}\,(1-t)^{b-1}\,dt \qquad (a, b > 0, 0 \le x \le 1))$$

Incomplete Beta Function (regularized)

$$I_x(a,b) = \frac{B(x;\, a, b)}{B(a,b)}$$

Incomplete Gamma Function (lower)

$$\gamma(s,x) = \int_0^x t^{s-1}\,e^{-t}\,dt$$

Incomplete Gamma Function (regularized)

$$P(s,x) = \frac{\gamma(s,x)}{\Gamma(s)}$$

Modified Bessel Function

$$I_\alpha(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(m+\alpha+1)}\left(\frac{x}{2}\right)^{2m+\alpha}$$

References

Chou, C., & H. Liu, (1998). "Properties of the half-normal distribution and its application to quality control", *Journal of Industrial Technology* Vol. 14(3) pp 4-7

Chou,Y., A.M. Polansky, and R.L. Mason (1998). "Transforming Non-Normal Data to Normality in Statistical Process Control," *Journal of Quality Technology*, Vol. 30(2), pp. 133-141.

David, H.A. (1981). *Order Statistics*, John Wiley & Sons, New York.

Giesbrecht, F. and A.H. Kempthorne (1966). "Maximum Likelihood Estimation in the Three-parameter Lognormal Distribution", *Journal of the Royal Statistical Society*, B 38, pp. 257-264.

W.H.Greene *Econometric Analysis* 4th Ed Prentice Hall, New Jersey.

Johnson, N. L., and S. Kotz, S. (1990). "Use of moments in deriving distributions and some characterizations", *Mathematical Scientist*, Vol. 15, pp. 42-52.

Johnson, N.L., S. Kotz, and N. Balakrishnan (1994) *Continuous Univariate Distributions*, Vol. 1, Wiley-Interscience.

Lockhart, R.A. and M.A. Stephens (1994)."Estimation and Tests of Fit for the Three-parameter Weibull Distribution", *Journal of the Royal Statistical Society*, Vol.56(3), pp. 491-500.

Nocedal, J. and S.J. Wright (1999). *Numerical Optimization*, Springer-Verlag, New York. Johnson, N.L. and S. Katz,(1970).*Distributions in Statistics: Continuous Uniuariate. Distributions–I*, Wiley, New York.

Tadikamalla,P.,R. (1980). "Notes and Comments: On Simulating Non-Normal Distributions", *Psychometrika*, Vol. 45(2), pp. 273-279.

Wang, J.Z. (2005). " A note on Estimation in the Four Paramter Beta Distribution", *Comm in Stats Simulation and computation*, Vol. 34 pp. 495-501.

Index B

Index