Financial Analysis Package MT

for GAUSS™

Version 2.0

Aptech Systems, Inc.

Documentation Version: May 1, 2003

Part Number: 003497

Contents

Chapter 1

Installation

1.1 UNIX

If you are unfamiliar with UNIX, see your system administrator or system documentation for information on the system commands referred to below. The device names given are probably correct for your system.

1.1.1 Download

1. Copy the `.tar.gz` file to `/tmp`.
2. Unzip the file.

   `gunzip` *appxxx*`.tar.gz`

3. `cd` to the **GAUSS** or **GAUSS Engine** installation directory. We are assuming `/usr/local/gauss` in this case.

   `cd /usr/local/gauss`

4. Untar the file.

   `tar xvf /tmp/`*appxxx*`.tar`

1.1.2 Floppy

1. Make a temporary directory.

   `mkdir /tmp/workdir`

2. `cd` to the temporary directory.

   `cd /tmp/workdir`

3. Use `tar` to extract the files.

   `tar xvf` *device_name*

   If this software came on diskettes, repeat the `tar` command for each diskette.

4. Read the README file.

   `more README`

5. Run the `install.sh` script in the work directory.

   `./install.sh`

   The directory the files are install to should be the same as the install directory of **GAUSS** or the **GAUSS Engine**.

6. Remove the temporary directory (optional).

The following device names are suggestions. See your system administrator. If you are using Solaris 2.x, see Section 1.1.3.

Operating System	3.5-inch diskette	1/4-inch tape	DAT tape
Solaris 1.x SPARC	`/dev/rfd0`	`/dev/rst8`	
Solaris 2.x SPARC	`/dev/rfd0a` (vol. mgt. off)	`/dev/rst12`	`/dev/rmt/1l`
Solaris 2.x SPARC	`/vol/dev/aliases/floppy0`	`/dev/rst12`	`/dev/rmt/1l`
Solaris 2.x x86	`/dev/rfd0c` (vol. mgt. off)		`/dev/rmt/1l`
Solaris 2.x x86	`/vol/dev/aliases/floppy0`		`/dev/rmt/1l`
HP-UX	`/dev/rfloppy/c20Ad1s0`		`/dev/rmt/0m`
IBM AIX	`/dev/rfd0`	`/dev/rmt.0`	
SGI IRIX	`/dev/rdsk/fds0d2.3.5hi`		

1.1.3 Solaris 2.x Volume Management

If Solaris 2.x volume management is running, insert the floppy disk and type

```
volcheck
```

to signal the system to mount the floppy.

The floppy device names for Solaris 2.x change when the volume manager is turned off and on. To turn off volume management, become the superuser and type

```
/etc/init.d/volmgt off
```

To turn on volume management, become the superuser and type

```
/etc/init.d/volmgt on
```

1.2 Windows/NT/2000

1.2.1 Download

Unzip the `.zip` file into the **GAUSS** or **GAUSS Engine** installation directory.

1.2.2 Floppy

1. Place the diskette in a floppy drive.
2. Call up a DOS window
3. In the DOS window log onto the root directory of the diskette drive. For example:

```
A:<enter>
cd\<enter>
```

4. Type: **ginstall** *source_drive target_path*

source_drive	Drive containing files to install with colon included For example: **A:**
target_path	Main drive and subdirectory to install to without a final \ For example: **C:\GAUSS**

A directory structure will be created if it does not already exist and the files will be copied over.

*target_path***src**	source code files
*target_path***lib**	library files
*target_path***examples**	example files

1.3 Differences Between the UNIX and Windows/NT/2000 Versions

- If the functions can be controlled during execution by entering keystrokes from the keyboard, it may be necessary to press *Enter* after the keystroke in the UNIX version.

- On the Intel math coprocessors used by the Windows/NT/2000 machines, intermediate calculations have 80-bit precision, while on the current UNIX machines, all calculations are in 64-bit precision. For this reason, **GAUSS** programs executed under UNIX may produce slightly different results, due to differences in roundoff, from those executed under Windows/NT/2000.

Chapter 2

Financial Analysis Package

written by

Ronald Schoenberg

This package provides procedures for the econometric analysis of financial data.

2.1 Getting Started

GAUSS 5.0.22+ is required to use these routines.

2.1.1 README Files

The file **README.fan** contains any last minute information on this module. Please read it before using the procedures in this module.

2.1.2 Setup

The **FANPACMT** library must be active in order to use the procedures in the *Financial Analysis Package*. Please make certain to include `fanpacmt` in the **LIBRARY** statement at the top of your program or command file. This will enable **GAUSS** to find the *Financial Analysis Procedures*.

```
library fanpacmt,pgraph;
```

If you plan to make any right hand references to the global variables (described in the *REFERENCE* sections), you also need the statement:

```
#include fanpacmt.ext;
```

Finally, to reset global variables in succeeding executions of the command file, the following instruction can be used:

```
clearSession;
```

This could be included with the above statements without harm and would ensure the proper definition of the global variables for all executions of the command file.

The version number of each module is stored in a global variable:

_fan_ver 3×1 matrix: the first element contains the major version number of the *Financial Analysis Package*, the second element the minor version number, and the third element the revision number.

If you call for technical support, please have the version number of your copy of this module on hand.

2.2 Modeling with FANPACMT

FANPACMT is a set of keyword commands and procedures for the estimation of parameters of time series models via the maximum likelihood method. The package is divided into two parts: (1) easy-to-program keyword commands which simplify the modeling process; and (2) **GAUSS** procedures, which can be called directly to perform the computations.

The **FANPACMT** keyword commands considerably simplify the work for the analysis of time series. For example, the following command file (which may also be entered interactively)

```
library fanpacmt,pgraph;
session test 'Analysis of 1996 Intel Stock Prices';
setDataset stocks;
setSeries intel;
estimate run1 garch(1,1);
estimate run2 arima(1,2,1);
showResults;
plotSeries;
plotCV;
```

replaces about a hundred lines of **GAUSS** code using procedures. See Chapter 4 for a description of the keyword commands.

Summary of Keyword Commands

clearSession	clears session from memory, resets global variables
constrainPDCovPar	sets **NLP** global for constraining covariance matrix of parameters to be positive definite
computeLogReturns	computes log returns from price data
computePercentReturns	computes percent returns from price data
estimate	estimates parameters of a time series model
forecast	generates a time series and conditional variance forecast
getCV	puts conditional variances or variance-covariance matrices into global vector **_fan_CV**
getCOR	puts conditional correlations into global variable **_fan_COR**
getEstimates	puts model estimates into global variable **_fan_Estimates**
getResiduals	puts unstandardized residuals into global vector
getSeriesACF	puts autocorrelations into global variable **_fan_ACF**
getSeriesPACF	puts partial autocorrelations into global variable **_fan_PACF**
getSession	retrieves a data analysis session
getSR	puts standardized residuals into global vector
plotCOR	plots conditional correlations
plotCSD	plots conditional standard deviations
plotCV	plots conditional variances
plotQQ	generates quantile-quantile plot
plotSeries	plots time series
plotSeriesACF	plots autocorrelations

plotSeriesPACF	plots partial autocorrelations
plotSR	plots standardized residuals
session	initializes a data analysis session
setAlpha	sets inference alpha level
setConstraintType	sets type of constraints on parameters
setCovParType	sets type of covariance matrix of parameters
setCVIndEqs	declares list of independent variables to be included in conditional variance equations
setDataset	sets dataset name
setIndEqs	declares list of independent variables to be included in mean equations
setInferenceType	sets type of inference
setIndVars	declares names of independent variables
setLagTruncation	sets lags included for FIGARCH model
setLagInitialization	sets lags excluded for FIGARCH model
setLjungBoxOrder	sets order for Ljung-Box statistic
setOutputFile	sets output file name
setRange	sets range of data
setSeries	declares names of time series
setVarNames	sets variable names for data stored in ASCII file
showEstimates	displays estimates in simple format
showResults	displays results of estimations
showRuns	displays runs
simulate	generates simulation
testSR	generates skew, kurtosis, Ljung-Box statistics

If the computations performed by the **FANPACMT** keyword commands do not precisely fit your needs, you may design your own command files using the **FANPACMT** procedures. For example, you may want to impose alternative sets of constraints on the parameters of a FIGARCH model. To do this you would design your own FIGARCH estimation using the **FANPACMT** procedures discussed in Section 2.9 in this chapter, and described in Chapter 4.

You might also want to write your own procedures for models not included in **FANPACMT**. To do this you will need to write a procedure for computing the log-likelihood and call the **SQPsolveMT** procedure for the estimation. These procedures are discussed in the **GAUSS** documention for the **Run-Time Library**.

When the **FANPACMT** keyword commands are used, analysis results are stored in a file on disk. This information can be retrieved or modified as necessary. Results are not stored if there is an error, and thus the original results are not lost when this happens. These keyword commands can be invoked either in command files or interactively from the **GAUSS** command line. They may also be mixed with other **GAUSS** commands either in a command file or interactively.

The following models are available in **FANPACMT**:

ols	normal linear regression model
$tols$	t distribution linear regression model
$arma(m, n)$	Normal ARMA model
$tarma(m, n)$	t distribution ARMA model
$garch(p, q)$	Normal GARCH model
$agarch(p, q)$	Normal GARCH model with asymmetry parameters
$tagarch(p, q)$	t distribution GARCH model with asymmetry
	parameters
$gtagarch(p, q)$	skew gen. t distribution GARCH model with asymmetry
	parameters
$tgarch(p, q)$	t distribution GARCH model
$gtgarch(p, q)$	skew gen. t distribution GARCH model
$igarch(p, q)$	Normal integrated GARCH model
$tigarch(p, q)$	t distribution integrated GARCH model
$gtigarch(p, q)$	skew gen. t distribution integrated GARCH model
$egarch(p, q)$	exponential GARCH model
$negarch(p, q)$	Normal GARCH model with leverage parameters
$tegarch(p, q)$	t distribution GARCH model with leverage
	parameters
$gtegarch(p, q)$	skew gen. t distribution GARCH model with leverage
	parameters
$figarch(p, q)$	Normal fractionally integrated GARCH model
$fitgarch(p, q)$	t distribution fractionally integrated GARCH model
$figtgarch(p, q)$	skew gen. t distribution fractionally integrated GARCH model
$varma(m, n)$	Normal multivariate VARMA model
$tvarma(m, n)$	t distribution multivariate VARMA model
$dvgarch(p, q)$	Normal DVEC multivariate GARCH model
$cdvgarch(p, q)$	constant correlation Normal DVEC multivariate GARCH model
$dvtgarch(p, q)$	t distribution DVEC multivariate GARCH model
$cdvtgarch(p, q)$	constant correlation t distribution DVEC multivariate GARCH model
$bkgarch(p, q)$	Normal BEKK multivariate GARCH model
$bktgarch(p, q)$	t distribution BEKK multivariate GARCH model

For an ARCH model set p = 0.

For ARMA-GARCH models use (p, q, m, n) where m is the order of the auto-regression parameters, and n the order of the moving average parameters.

2.3 Univariate Time Series Models

2.3.1 ARCH/GARCH/ARMAGARCH models

For the generalized autoregressive conditional heteroskedastic (GARCH) model let

$$\Theta(L)\epsilon_t = \Phi(L)y_t - x_t\beta - \delta\sigma_t$$

where $t = 1, 2, ...T$, and y_t an observed time series, x_t an observed time series of fixed exogenous variables including a column of ones, β a vector of coefficients, $\Theta(L) = 1 + \theta_1 L + \theta_2 L^2 \ldots + \theta_n L^n$, $\Phi(L) = 1 - \phi_1 L - \phi_2 L^2 \ldots - \phi_m L^m$, and $\sigma_t = E(\epsilon_t)$.

Also define

$$\epsilon_t \equiv \eta_t \sigma_t$$

where $E(\eta_t) = 0$, $Var(\eta_t) = 1$.

standard model

The standard specification of the conditional variance is

$$\begin{aligned} \sigma_t^2 &= \omega + \alpha_1\epsilon_{t-1}^2 + \cdots + \alpha_q\epsilon_{t-q}^2 + \\ &\quad \beta_1\sigma_{t-1}^2 + \cdots + \beta_p\sigma_{t-p}^2 + \Gamma Z_t \end{aligned}$$

where Z_t is an observed time series of fixed exogenous variables.

There are two variations of the specification of the conditional variance, the “leverage” or “egarch” model (Nelson,1991) and the “asymmetry” or “agarch” model (Glosten, L.R., Jagannathan, R., and Runkle, D.E., 1993).

Leverage Model

$$\begin{aligned} log\sigma_t^2 &= \omega + \alpha_1(|\epsilon_{t-1}| - E(|\epsilon_{t-1}|) + \zeta\epsilon_{t-1}) \cdots + \alpha_q(|\epsilon_{t-q}| - E(|\epsilon_{t-q}|) + \zeta\epsilon_{t-q}) + \\ &\quad \beta_1\sigma_{t-1}^2 + \cdots + \beta_p\sigma_{t-p}^2 + \Gamma Z_t \end{aligned}$$

Asymmetry Model

$$\begin{aligned} \sigma_t^2 &= \omega + (\alpha_1 + \tau S_{t-1})\epsilon_{t-1}^2 + \cdots + (\alpha_q + \tau S_{t-q})\epsilon_{t-q}^2 + \\ &\quad \beta_1\sigma_{t-1}^2 + \cdots + \beta_p\sigma_{t-p}^2 + \Gamma Z_t \end{aligned}$$

where $S_t = 1$ if $\epsilon_t < 0$ and $S_t = 0$ otherwise.

log-likelihood

For maximum likelihood estimation of all models, we provide two distributions for η_t, the Normal and Student's t, and in addition to these for the EGARCH there is also the generalized exponential distribution. For some univariate distributions the skew generalized t distribution is provided.

The log-likelihood conditional on $\mu = max(p,q)$ initial estimates of the conditional variances is, for $\eta_t \sim N(0,1)$

$$logL = -\frac{T-\mu}{2}log(2\pi) - \sum_{t+\mu+1}^{T} log(\sigma_t) - \frac{1}{2}\sum_{t+\mu+1}^{T}\frac{\epsilon_t^2}{\sigma_t^2}$$

where

$$\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_\mu^2 = \frac{1}{T}\sum_{t=1}^{T}\epsilon_t^2$$

Student's t distribution

The unit t distribution with ν degrees of freedom and variance σ^2 for η_t is

$$f(\eta_t) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)(\nu-2)^{1/2}\pi^{1/2}\sigma}\left(1+\frac{\eta_t^2}{(\nu-2)\sigma^2}\right)^{-(\nu+1)/2)}$$

The conditional log-likelihood for $\eta_t \sim t(0,1,\nu)$ is then

$$\begin{aligned} logL \;\; = \;\; & -\tfrac{T-\mu}{2}log\left(\tfrac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)(\nu-2)^{1/2}\pi^{1/2}\sigma}\right) - \textstyle\sum_{t+\mu}^{T} log(\sigma_t) \\ & -\tfrac{\nu+1}{2}\textstyle\sum_{t+\mu}^{T} log\left(1+\tfrac{\epsilon_t^2}{(\nu-2)\sigma_t^2}\right) \end{aligned}$$

Generalized Error Distribution

For the generalized error distribution

$$f(\eta_t) = \frac{\rho\Gamma(3/\rho)^{\frac{1}{2}}}{2\sigma_t^2\Gamma(1/\rho)^{\frac{3}{2}}}e^{-\frac{1}{2}|\frac{\eta_t}{\lambda\sigma_t}|^{\rho}}$$

where $\rho > 0$ is a parameter measuring the thickness of the tails, δ is a *leverage* parameter,

$$\lambda = 2^{-1/\rho}\Gamma(1/\rho)^{\frac{1}{2}}\Gamma(3/\rho)^{-\frac{1}{2}}$$

and

$$E|\eta_t| = \Gamma(2/\rho)^{\frac{1}{2}}\Gamma(1/\rho)^{-\frac{1}{2}}\Gamma(3/\rho)^{-\frac{1}{2}}$$

The log-likelihood is

$$logL = log(\frac{\rho}{2}) + \frac{1}{2}log\Gamma(3/\rho) - \frac{3}{2}log\Gamma(1/\rho) - \frac{1}{2}\sum_{t+\kappa}^{T}\left|\frac{\eta^2}{\lambda\sigma_t}\right|^{\rho} - \sum_{t+\kappa}^{T}\sigma_t^2$$

where $\kappa = max(p,q)+1$.

FANPACMT

Skew Generalized T Distribution

The skew generalized t distribution is described in Theodossiou (1998). The log-likelihood for an observation is

$$logl_i = log\gamma_i - \frac{\nu+1}{\kappa} log \left[1 + \left(\frac{|\frac{u_i+\mu_i}{\sigma_i}|}{\theta(1+\lambda sign(u_i+\mu_i)} \right) \right]$$

where

$$\begin{aligned}
\gamma_i &= 0.5S\sigma_i^{-1}\kappa B(1/\kappa, \nu/\kappa)^{-\frac{3}{2}} B(3/\kappa, (\nu-2)/\kappa)^{\frac{1}{2}} \\
\theta &= S^{-1} B(1/\kappa, \nu/\kappa)^{\frac{1}{2}} B(3/\kappa, (\nu-2)/\kappa)^{-\frac{1}{2}} \\
S &= \left(1 + 3\lambda^2 + 4\lambda^2 B(2/\kappa, (\nu-1)/\kappa)^2 B(1/\kappa, \nu/\kappa)^{-1} B(3/\kappa, (\nu-2)/\kappa)^{-1}\right)^{\frac{1}{2}} \\
\mu_i &= 2\sigma_i \lambda B(2/\kappa, (\nu-1)/\kappa) B(1/\kappa, \nu/\kappa)^{-\frac{1}{2}} B(3/\kappa, (\nu-2)/\kappa)^{-\frac{1}{2}}
\end{aligned}$$

and where $-1 < \lambda < 1$ is a skew parameter, and $\kappa > 0$ and $\nu > 2$ are kurtosis parameters. For $\lambda = 0$ and $\kappa = 2$ we have the Student's t distribution, for $\lambda = 0, \kappa = 1, \nu = \inf$ we have the Laplace distribution, for $\lambda = 0, \kappa = 2, \nu = \inf$ the Normal distribution, and for $\lambda = 0, \kappa = \inf, \nu = \inf$ the uniform distribution.

Nonnegativity of Conditional Variances

Constraints may be placed on the parameters to enforce the stationarity of the GARCH model as well as the nonnegativity of the conditional variances.

Nelson and Cao (1992) established necessary and sufficient conditions for nonnegativity of the conditional variances for the GARCH(1,q) and GARCH(2,q) models.

GARCH(1,q).

$$\omega \geq 0$$

$$\beta_1 \geq 0$$

$$\sum_{j=0}^{k} \alpha_{j+1}\beta^{k-j} \geq 0, \; k = 0, \cdots, q-1$$

GARCH(2,q). Define Δ_1 and Δ_2 as the roots of

$$1 - \beta_1 Z^{-1} - \beta_2 Z^{-2}$$

Then

$$\omega/(1 - \Delta_1 - \Delta_2 + \Delta_1\Delta_2) \geq 0$$

$$\beta 1^2 + 4\beta_2 \geq 0$$

$$\Delta_1 > 0$$

$$\sum_{j=0}^{q-1} \alpha_{j+1}\Delta^{-j} > 0$$

and,

$$\phi_k \geq 0, k = 0, \cdots, q$$

where

$$\begin{aligned}
\phi_0 &= \alpha_1 \\
\phi_1 &= \beta_1\phi_0 + \alpha_2 \\
\phi_2 &= \beta_1\phi_1 + \beta_2\phi_0 + \alpha_3 \\
&\cdot \\
&\cdot \\
&\cdot \\
\phi_q &= \beta_1\phi_{q-1} + \beta_2\phi_{q-2}
\end{aligned}$$

GARCH(p,q). General constraints for $p > 2$ haven't been worked out. For such models, **FANPACMT** directly constrains the conditional variances to be greater than zero. It also constrains the roots of the polynomial

$$1 - \beta_1 Z - \beta_2 Z^2 - \cdots - \beta_p Z^p$$

to be outside the unit circle. This only guarantees that the conditional variances will be nonnegative in the sample, and does not guarantee that the conditional variances will be nonnegative for all realizations of the data.

Stationarity

To ensure that the GARCH process is covariance stationary, the roots of

$$1 - (\alpha_1 + \beta_1)Z - (\alpha_2 + \beta_2)Z^2 - \cdots$$

may be constrained to be outside the unit circle (Gouriéroux, 1997, page 37).

Most GARCH models reported in the economics literature are estimated using software that cannot impose nonlinear constraints on parameters and thus either impose a more highly restrictive set of linear constraints than the ones described here, or impose no constraints at all. The procedures provided in **FANPACMT** ensure that you have the best fitting solution that satisfies the conditions of stationarity and nonnegative of conditional variances.

Stationarity in the GARCH-in-cv model is conditional on the exogenous variables included in the conditional variance equation. There is no assurance of unconditional stationarity without further constraints or assumptions with respect to the exogenous variables.

Initialization

The calculation of the log-likelihood is recursive and requires initial values for the conditional variance. Following standard practice, the first q values of the conditional variances are fixed to the sample unconditional variance of the series.

2.3.2 IGARCH

The IGARCH(p,q) model is a GARCH(p,q) model with a unit root. This is accomplished in **FANPACMT** by adding the equality constraint

$$\sum_i \alpha_1 + \sum_i \beta_i = 1$$

2.3.3 FIGARCH

Define the time series

$$\epsilon_t = y_t - x_t\beta$$

where $t = 1, 2, ...T$, and y_t an observed time series, x_t an observed time series of exogenous variables including a column of ones, and β a vector of coefficients.

Further define

$$\epsilon_t \equiv \eta_t\sigma_t$$

where $E(\eta_t) = 0$, $Var(\eta_t) = 1$.

Let

$$A(L) = \alpha_1 L + \alpha_2 L^2 + \cdots + \alpha_q L^q$$

and

$$A(L) = \beta_1 L + \beta_2 L^2 + \cdots + \beta_p L^p$$

where L is the lag operator. In this notation, the GARCH(p,q) model can be specified

$$\sigma_t^2 = \omega + A(L)\epsilon_t^2 + B(L)\sigma_t^2$$

The GARCH(p,q) model can be re-specified as an ARMA(max(p,q),p) model (Bailie, et al., 1996)

$$[1 - A(L) - B(L)]\epsilon_t^2 = \omega + [1 - B(L)]\nu_t$$

where $\nu_t \equiv \epsilon_t^2 - \sigma_t^2$ is the "innovation" at time t for the conditional variance process.

Using this notation, the IGARCH(p,q) model is

$$\theta(L)(1 - L)\epsilon_t^2 = \omega + [1 - B(L)]\nu_t$$

where $\theta(L) = [1 - A(L) - B(L)](1 - L)^{-1}$. The *fractionally integrated* GARCH or FIGARCH(p,q) model is

$$\theta(L)(1 - L)^d\epsilon_t^2 = \omega + [1 - B(L)]\nu_t$$

where $0 < d < 1$.

FIGARCH-in-cv

For the FIGARCH-in-cv model, independent variables may be added to the conditional variance equation

$$\sigma_t^2 = \omega + A(L)\epsilon_t^2 + B(L)\sigma_t^2 + Z_t\Gamma$$

where Z_t is the t-th vector of observed independent variables and Γ a matrix of coefficients.

log-likelihood

The conditional variance in the FIGARCH(p,q) model is the sum of an infinite series of prior conditional variances:

$$\begin{aligned}\sigma_t^2 &= \omega + [1 - B(L) - \theta(L)(1-L)^d]\epsilon_t^2 + B(L)\sigma_t^2 \\ &= \omega + [1 - B(L) - [1 - A(L) - B(L)](1-L)^{d-1}]\epsilon_t^2 + B(L)\sigma_t^2 \\ &= \omega + (\phi_1 L - \phi_2 L^2 - \cdots)\epsilon_t^2 + B(L)\sigma_t^2\end{aligned}$$

$$\phi_k = \alpha_k - \pi_k + \sum_{i=1}^{k-1} \pi_i(\alpha_{k-i} + \beta_{k-i})$$

where $\alpha_j = 0, j > q$ and $\beta_j = 0, j > p$, and

$$\pi_k = \frac{1}{k!}\prod_{i=1}^{k}(i-d)$$

In practice, the log-likelihood will be computed from available data and this means that the calculation of the conditional variance will be truncated. To minimize this error, the log-probabilities for initial observations can be excluded from the log-likelihood. The default is one half of the observations.

This can be modified by calling the keyword command **setLagTruncation** with an argument specifying the number of observations to be *included* in the log-likelihood, or by directly setting the **FANPACMT** global, **_fan_init** to the number of initial observations to be *excluded* from the log-likelihood.

The log-likelihood for $\eta_t \sim N(0,1)$ is

$$logL = -\frac{T-\rho}{2}log(2\pi) - \sum_{t+\mu}^{T} log(\sigma_t) - \frac{1}{2}\sum_{t+\mu}^{T}\frac{\epsilon_t^2}{\sigma_t^2}$$

and for $\eta_t \sim t(0,1,\nu)$ is

$$\begin{aligned}logL &= -\tfrac{T-q}{2}log\left(\tfrac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)(\nu-2)^{1/2}\pi^{1/2}\sigma}\right) - \textstyle\sum_{t=q}^{T} log(\sigma_t) \\ &\quad -\tfrac{\nu+1}{2}\textstyle\sum_{t=q}^{T} log\left(1 + \tfrac{\epsilon_t^2}{(\nu-2)\sigma_t^2}\right)\end{aligned}$$

where μ = **_fan_init**, the number of lags used to initialize the process.

Stationarity

The unconditional variance of FIGARCH models is infinite, and thus is not covariance stationary. However, Baillie, et al. (1996) point out that FIGARCH models are ergodic and strictly stationary for $0 \leq d \leq 1$ using a direct extension of proofs for the IGARCH case (Nelson, 1990).

In addition to the constraint on d, it is also necessary to constrain the roots of

$$1 - \beta_1 Z - \beta_2 Z^2 - \cdots - \beta_p Z^p$$

to be outside the unit circle.

Nonnegative conditional variances

General methods to ensure the nonnegativity of the conditional variances haven't been established. However, in **FANPACMT** the conditional variances are directly constrained to be nonnegative. This guarantees nonnegative conditional variances in the sample, but does not do so for all realizations of the time series.

Stationarity in the FIGARCH-in-cv model is conditional on the exogenous variables included in the conditional variance equation. There is no assurance of unconditional stationarity without further constraints or assumptions with respect to the exogenous variables.

2.3.4 OLS

Define the series

$$\epsilon_t = y_t - x_t \beta$$

where $t = 1, 2, ...T$, and y_t an observed time series, x_t an observed time series of exogenous variables including a column of ones, and β a vector of coefficients.

For $\epsilon_t \sim N(0, 1)$, the ordinary least squares estimator

$$\hat{\beta} = (X'X)^{-1} X'Y$$

where x_t' and y_t are the i-th rows of X and Y, respectively, is maximum likelihood.

For ϵ_t with a t distribution with ν degrees of freedom and variance σ^2, the log-likelihood is

$$logL = -\frac{T}{2} log \left(\frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)(\nu-2)^{1/2}\pi^{1/2}\sigma} \right) - \sum_{t=q}^{T} log(\sigma) - \frac{\nu+1}{2} \sum_{t=q}^{T} log \left(1 + \frac{\epsilon_t^2}{(\nu-2)\sigma^2} \right)$$

It is also necessary to constrain ν to be greater than 2.

2.4 Multivariate Time Series Models

2.4.1 Diagonal Vec ARCH/GARCH/VARGARCH

For the Diagonal Vec VARGARCH model let

$$\Theta(L)\epsilon_t = \Phi(L)y_t - x_t\beta - \delta\sigma_t$$

where $t = 1, 2, ...T$, and y_t is a vector of observed time series, x_t an observed time series of fixed exogenous variables including a column of ones, β a matrix of coefficients, $\Theta(L) = 1 + \theta_1 L + \theta_2 L^2 \ldots + \theta_n L^n$, $\Phi(L) = 1 - \phi_1 L - \phi_2 L^2 \ldots - \phi_m L^m$, and $\sigma_t = diag(E(\epsilon_t))$.

Each nonredundant element of Σ_t is a separate GARCH model

$$\begin{aligned}\Sigma_{t,ij} &= \Omega_{ij} + A_{1,ij}\epsilon_{i,t-1}\epsilon_{j,t-1} + \cdots + A_{q,ij}\epsilon_{i,t-q}\epsilon_{j,t-q} \\ &\quad B_{1,ij}\Sigma_{t,ij-1} + \cdots + B_{p,ij}\Sigma_{t,ij-p}\end{aligned}$$

where $\Omega_{ij} = \Omega_{ji}$ and $A_{k,ij} = A_{k,ji}$.

DVEC GARCH-in-cv

For the DVEC GARCH-in-cv (or DVGARCHV) model, independent variables are added to the equation for the conditional variance

$$\begin{aligned}\Sigma_{t,ij} &= \Omega_{ij} + A_{1,ij}\epsilon_{i,t-1}\epsilon_{j,t-1} + \cdots + A_{q,ij}\epsilon_{i,t-q}\epsilon_{j,t-q} \\ &\quad +B_{1,ij}\Sigma_{t,ij-1} + \cdots + B_{p,ij}\Sigma_{t,ij-p} + Z_t\Gamma_{ij}\end{aligned}$$

where Z_t is the t-th vector of observed independent variables and Γ_{ij} a matrix of coefficients.

DVEC GARCH-in-mean

For the DVEC GARCH-in-mean (or DVGARCHM) model, the time series equation is modified to include the conditional variance

$$\epsilon_{i,t} = y_{i,t} - x_t\beta_i' - \delta_i\Sigma_{t,ii}$$

where β_i is the i-th row of B, an $\ell \times k$ coefficient matrix.

log-likelihood

The log-likelihood conditional on $\mu = max(p,q)$ initial estimates of the conditional variance-covariances matrices is

$$logL = -\frac{k(T-\mu)}{2}log(2\pi) - \frac{1}{2}\sum_{t+\mu+1}^{T} log \mid \Sigma_t \mid -\frac{1}{2}\sum_{t+\mu+1}^{T}(\epsilon_t'\Sigma_t^{-1}\epsilon_t)$$

where

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_\mu = \frac{1}{T}\sum_{t=1}^{T}\epsilon_t'\epsilon_t$$

The conditional log-likelihood for a t-distributed ϵ is

$$\begin{aligned} logL &= -\tfrac{T-\mu}{2}(log\Gamma((\nu+k)/2) - log\Gamma(\nu/2) - \tfrac{1}{2}log((\nu-2)\pi)) \\ &\quad -\tfrac{1}{2}\textstyle\sum_{t+\mu+1}^{T} log \mid \Sigma_t \mid -\tfrac{\nu+k}{2}\textstyle\sum_{t+\mu+1}^{T} log(1+\epsilon_t'\Sigma_t^{-1}\epsilon_t/(\nu-2)) \end{aligned}$$

Positive Definiteness of Conditional Variances

Constraints on the parameters are necessary to enforce the positive definiteness of the conditional variance-covariances matrices. This requirement is assured by directly constraining the eigenvalues of the conditional variance-covariance matrices to be greater than zero.

Stationarity

Stationarity is assured if the roots of the determinantal equation

$$\mid I - (A_1 + B_1)z - (A_2 + B_2)z^2 - ... \mid$$

lie outside the unit circle (Gourieroux, 1997).

Stationarity in the DVEC GARCH-in-cv model is conditional on the exogenous variables included in the conditional variance equation. There is no assurance of unconditional stationarity without further constraints or assumptions with respect to the exogenous variables.

2.4.2 Constant Correlation DVEC GARCH Model

Define a vector of ℓ residuals

$$\epsilon_t = y_t - x_t\beta$$

where $t = 1, 2, ...T$, and y_t an observed multiple time series, x_t an observed time series of exogenous variables including a column of ones, and β a matrix of coefficients.

Let Σ_t be the conditional variance-covariance matrix of ϵ_t with constant correlation matrix R. Then each diagonal element of Σ_t is modeled as a separate GARCH model

$$\Sigma_{t,ii} = \Omega_i + A_{i,1}\epsilon^2_{i,t-1} + \cdots + A_{i,q}\epsilon^2_{i,t-q} \\ +B_{i,1}\Sigma_{i,t-1} + \cdots + B_{i,p}\Sigma_{i,t-p}$$

The elements of the conditional variance-covariance matrix, then, are

$$\Sigma_{t,ij} = R_{ij}\sqrt{\Sigma_{t,ii}\Sigma_{t,jj}}$$

constant correlation DVEC GARCH-in-cv

For the constant correlation DVEC GARCH-in-cv (or CDVGARCHV) model, independent variables are added to the equation for the conditional variance

$$\Sigma_{t,ii} = \Omega_i + A_{i,1}\epsilon^2_{i,t-1} + \cdots + A_{i,q}\epsilon^2_{i,t-q} \\ +B_{i,1}\Sigma_{i,t-1} + \cdots + B_{i,p}\Sigma_{i,t-p} + Z_t\Gamma_{ij}$$

where Z_t is the t-th vector of observed independent variables and Γ_{ij} a matrix of coefficients.

constant correlation DVEC GARCH-in-mean

For the constant correlation DVEC GARCH-in-mean (or DVGARCHM) model, the time series equation is modified to include the conditional variance

$$\epsilon_{i,t} = y_{i,t} - x_t\beta_i' - \delta_i\Sigma_{t,ii}$$

where β_i is the i-th row of B, an $\ell \times k$ coefficient matrix.

FANPACMT

log-likelihood

The log-likelihood conditional on $\mu = max(p,q)$ initial estimates of the conditional variance-covariances matrices is

$$logL = -\frac{k(T-\mu)}{2}log(2\pi) - \frac{1}{2}\sum_{t+\mu+1}^{T} log \mid \Sigma_t \mid - \frac{1}{2}\sum_{t+\mu+1}^{T}(\epsilon_t' \Sigma_t^{-1}\epsilon_t)$$

where

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_\mu = \frac{1}{T}\sum_{t=1}^{T}\epsilon_t'\epsilon_t$$

and where

$$\sigma_{t,ij} = r_{ij}\sqrt{\sigma_{t,ii}\Sigma_{t,jj}}$$

where r_{ij} is a constant parameter to be estimated.

The conditional log-likelihood for a t-distributed ϵ is

$$\begin{aligned} logL \quad = \quad & -\tfrac{T-\mu}{2}(log\Gamma((\nu+k)/2) - log\Gamma(\nu/2) - \tfrac{1}{2}log((\nu-2)\pi)) \\ & -\tfrac{1}{2}\textstyle\sum_{t+\mu+1}^{T} log \mid \Sigma_t \mid - \tfrac{\nu+k}{2}\textstyle\sum_{t+\mu+1}^{T} log(1+\epsilon_t'\Sigma_t^{-1}\epsilon_t/(\nu-2)) \end{aligned}$$

Positive Definiteness of Conditional Variances

Constraints on the parameters are necessary to enforce the positive definiteness of the conditional variance-covariances matrices. This requirement is assured by directly constraining the eigenvalues of the conditional variance-covariance matrices to be greater than zero.

Stationarity

Stationarity is assured if the roots of the determinantal equation (Gourieroux, 1997)

$$\mid I - (A_1 + B_1)z - (A_2 + B_2)z^2 - \cdots \mid$$

lie outside the unit circle. Since the A_i and B_i are diagonal matrices, this amounts to determining the roots of k polynomials

$$1 - (A_{111} + B_{111})z - (A_{211} + B_{211})z^2 - \cdots \tag{2.1}$$
$$1 - (A_{122} + B_{122})z - (A_{222} + B_{222})z^2 - \cdots \tag{2.2}$$
$$\vdots \tag{2.3}$$
$$1 - (A_{1kk} + B_{1kk})z - (A_{2kk} + B_{2kk})z^2 - \cdots \tag{2.4}$$

Stationarity in the constant correlation DVEC GARCH-in-cv model is conditional on the exogenous variables included in the conditional variance equation. There is no assurance of unconditional stationarity without further constraints or assumptions with respect to the exogenous variables.

2.4.3 BEKK GARCH

Define a vector of ℓ residuals

$$\epsilon_t = y_t - x_t\beta$$

where $t = 1, 2, ...T$, and y_t an observed multiple time series, x_t an observed time series of exogenous variables including a column of ones, and β a matrix of coefficients.

Further define the conditional variance Σ_t of ϵ_t

$$\begin{aligned}\Sigma_t &= \Omega + A_1\epsilon'_{t-1}\epsilon_{t-1}A'_1 + \cdots + A_q\epsilon'_{t-q}\epsilon_{t-q}A'_q + \\ &\quad B_1\Sigma_{t-1}B'_1 + \cdots + B_p\Sigma_{t-p}B'_p\end{aligned}$$

where Ω is a symmetric matrix, and A_i and B_i are square matrices.

log-likelihood

The log-likelihood conditional on $\mu = max(p, q)$ initial estimates of the conditional variance-covariances matrices is

$$logL = -\frac{k(T-\mu)}{2}log(2\pi) - \frac{1}{2}\sum_{t+\mu+1}^{T} log \mid \Sigma_t \mid -\frac{1}{2}\sum_{t+\mu+1}^{T}(\epsilon'_t\Sigma_t^{-1}\epsilon_t)$$

where

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_\mu = \frac{1}{T}\sum_{t=1}^{T}\epsilon'_t\epsilon_t$$

The conditional log-likelihood for a t-distributed ϵ is

$$\begin{aligned}logL &= -\tfrac{T-\mu}{2}(log\Gamma((\nu+k)/2) - log\Gamma(\nu/2) - \tfrac{1}{2}log((\nu-2)\pi)) \\ &\quad -\tfrac{1}{2}\textstyle\sum_{t+\mu+1}^{T} log \mid \Sigma_t \mid -\tfrac{\nu+k}{2}\textstyle\sum_{t+\mu+1}^{T} log(1 + \epsilon'_t\Sigma_t^{-1}\epsilon_t/(\nu-2))\end{aligned}$$

Positive Definiteness of Conditional Variances

Constraints on the parameters are necessary to enforce the positive definiteness of the conditional variance-covariances matrices. This requirement is assured by directly constraining the eigenvalues of the conditional variance-covariance matrices to be greater than zero.

FANPACMT

2.5 Inference

The parameters of time series models in general are highly constrained. This presents severe difficulties for statistical inference. The usual method for statistical inference, comprising the calculation of the covariance matrix of the parameters and constructing t-statistics from the standard errors of the parameters, fails in the context of inequality constrained parameters because confidence regions will not generally be symmetric about the estimates. For this reason **FANPACMT** does not compute t-statistics, but rather computes and reports confidence limits.

The most common type of inference is based on the Wald statistic. A $(1 - \alpha)$ joint Wald-type confidence region for θ is the hyper-ellipsoid

$$JF(J, N-K; \alpha) = (\theta - \hat{\theta})' V^{-1} (\theta - \hat{\theta}), \tag{2.5}$$

where V is the covariance matrix of the parameters. The confidence limits are the maximum and minimum solution of

$$\min \left\{ \eta_k' \theta \mid (\theta - \hat{\theta})' V^{-1} (\theta - \hat{\theta}) \geq JF(J, N-K; \alpha)) \right\}, \tag{2.6}$$

where η can be an arbitrary vector of constants and $J = \sum \eta_k \neq 0$.

When there are no constraints, the solution to this problem for a given parameter is the well known

$$\hat{\theta} \pm t_{(1-\alpha)/2, T-k} \sigma_{\hat{\theta}}$$

where $\sigma_{\hat{\theta}}$ is the square root of the diagonal element of V associated with $\hat{\theta}$.

When there are constraints in the model, two things happen that render the classical method invalid. First, the solution to (2.6) is no longer (2.5) and second, (2.5) is not valid whenever the hyper-ellipsoid is on or near a constraint boundary.

(2.5) is based on an approximation to the likelihood ratio statistic. This approximation fails in the region of constraint boundaries because the likelihood ratio statistic itself is known to be distributed there as a *mixture* of chi-squares (Gouriéroux, et al.; 1982, Wolak, 1991). In finite samples these effects occur in the *region* of the constraint boundary, specifically when the true value is within $\epsilon = \sqrt{(\sigma_e^2/N)\chi^2_{(1-\alpha,k)}}$ of the constraint boundary.

Here, and in **FANPACMT**, we consider only the solution for a given parameter, a "parameter of interest;" all other parameters are "nuisance parameters." There are three cases to consider:

(1) parameter constrained, no nuisance parameters constrained;

(2) parameter unconstrained, one or more nuisance parameters constrained;

(3) parameter constrained, one or more nuisance parameters constrained.

Case 1: When the true value is on the boundary, the statistics are distributed as a simple mixture of two chi-squares. Monte Carlo evidence presented Schoenberg (1997) shows that this holds as well in finite samples for true values within ϵ of the constraint boundary.

Case 2: The statistics are distributed as weighted mixtures of chi-squares when the correlation of the constrained nuisance parameter with the unconstrained parameter of interest is greater than about .8. A correction for these effects is feasible. However, for finite samples, the effects on the statistics due to a true value of a constrained nuisance parameter being within ϵ of the boundary are greater and more complicated than the effects of actually being on the constraint boundary. There is no systematic strategy available for correcting for these effects.

Case 3: The references disagree. Gouriéroux, et al., (1982) and Wolak (1991) state that the statistics are distributed as a mixture of chi-squares. However, Self and Liang (1987) argue that when the distributions of the parameter of interest and the nuisance parameter are correlated, the distributions of the statistics are not chi-square mixtures.

There is no known solution for these problems with the type of confidence limits discussed here. Bayesian limits produce correct limits (Geweke, 1995), but they are considerably more computationally intensive. With the correction described in Schoenberg (1997), however, confidence limits computed via the inversion of the Wald statistic will be correct provided that no nuisance parameter within ϵ of a constraint boundary is correlated with the parameter of interest by more than about .6.

2.5.1 Covariance Matrix of Parameters

FANPACMT computes a covariance matrix of the parameters that is an approximate estimate when there are constrained parameters in the model (Gallant, 1987, Wolfgang and Hartwig, 1995). When the model includes inequality constraints, the covariance matrix computed directly from the Hessian, the usual method for computing this covariance matrix, is incorrect because they do not account for boundaries placed on the distributions of the parameters by the inequality constraints.

An argument based on a Taylor-series approximation to the likelihood function (e.g., Amemiya, 1985, page 111) shows that

$$\hat{\theta} \rightarrow N(\theta, A^{-1}BA^{-1}),$$

where

$$\begin{aligned} A &= E\left[\frac{\partial^2 L}{\partial\theta\partial\theta'}\right], \\ B &= E\left[\left(\frac{\partial L}{\partial\theta}\right)'\left(\frac{\partial L}{\partial\theta}\right)\right]. \end{aligned}$$

FANPACMT

Estimates of A and B are

$$\hat{A} = \frac{1}{N}\sum_{i}^{N}\frac{\partial^2 L_i}{\partial\theta\partial\theta'},$$

$$\hat{B} = \frac{1}{N}\sum_{i}^{N}\left(\frac{\partial L_i}{\partial\theta}\right)'\left(\frac{\partial L_i}{\partial\theta}\right).$$

Assuming the correct specification of the model plim(A) = plim(B),

$$\hat{\theta} \rightarrow N(\theta, \hat{A}^{-1}).$$

Without loss of generality we may consider two types of constraints: the nonlinear equality, and the nonlinear inequality constraints (the linear constraints are included in nonlinear, and the bounds are regarded as a type of linear inequality). Furthermore, the inequality constraints may be treated as equality constraints with the introduction of "slack" parameters into the model:

$$H(\theta) \geq 0$$

is changed to

$$H(\theta) = \zeta^2,$$

where ζ is a conformable vector of slack parameters.

Further, we distinguish *active* from *inactive* inequality constraints. Active inequality constraints have nonzero Lagrangeans, γ_j, and zero slack parameters, ζ_j, while the reverse is true for inactive inequality constraints. Keeping this in mind, define the diagonal matrix, Z, containing the slack parameters, ζ_j, for the inactive constraints, and another diagonal matrix, Γ, containing the Lagrangean coefficients. Also, define $H_\oplus(\theta)$ representing the active constraints, and $H_\ominus(\theta)$ the inactive.

The likelihood function augmented by constraints is then

$$\begin{aligned} L_A &= L + \lambda_1 g(\theta)_1 + \cdots + \lambda_I g(\theta)^I + \gamma_1 h_{\oplus 1}(\theta) + \cdots + \gamma_J h_{\oplus J}(\theta) \\ &\quad + h_{\ominus 1}(\theta)_i - \zeta_1^2 + \cdots + h_{\ominus K}(\theta) - \zeta_K^2, \end{aligned}$$

and the Hessian of the augmented likelihood is

$$E\left(\frac{\partial^2 L_A}{\partial\theta\partial\theta'}\right) = \begin{bmatrix} \Sigma & 0 & 0 & \dot{G}' & \dot{H}'_\oplus & \dot{H}'_\ominus \\ 0 & 2\Gamma & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2Z \\ \dot{G} & 0 & 0 & 0 & 0 & 0 \\ \dot{H}_\oplus & 0 & 0 & 0 & 0 & 0 \\ \dot{H}_\ominus & 0 & 2Z & 0 & 0 & 0 \end{bmatrix},$$

where the dot represents the Jacobian with respect to θ, $L = \sum_{i=1}^{N} \log P(Y_i; \theta)$, and $\Sigma = \partial^2 L/\partial\theta\partial\theta'$. The covariance matrix of the parameters, Lagrangeans, and slack parameters is the Moore-Penrose inverse of this matrix.

Construct the partitioned array

$$\tilde{B} == \begin{bmatrix} \dot{G} \\ \dot{H}_{\oplus} \\ \dot{H}_{\ominus} \end{bmatrix}.$$

Let Ξ be the orthonormal basis for the null space of $\tilde{B}$, then the covariance matrix of the parameters is

$$\Xi(\Xi'\Sigma\Xi)^{-1}\Xi'.$$

Rows of this matrix associated with active inequality constraints may not be available, i.e., the rows and columns of Ω associated with those parameters may be all zeros.

2.5.2 Quasi-Maximum Likelihood Covariance Matrix of Parameters

FANPACMT computes a QML covariance matrix of the parameters when requested. Define $B = (\partial L_A/\partial\theta)'(\partial L_A/\partial\theta)$ evaluated at the estimates. Then the covariance matrix of the parameters is $\Omega B\Omega$.

To request the QML covariance matrix, call the keyword command

```
setCovParType QML
```

The default ML covariance matrix can be set by

```
setCovParType ML
```

2.5.3 Confidence Limits

FANPACMT computes, by default, confidence limits computed in the standard way from t-statistics. These limits suffer from the deficiencies reported in the previous section – they are symmetric about the estimate, which is not usually the case for constrained parameters, and they can include undefined regions of the parameter space.

2.6 FANPACMT Keyword Commands

Summary of Keyword Commands

clearSession	clears session from memory, resets global variables
constrainPDCovPar	sets **NLP** global for constraining covariance matrix of parameters to be positive definite
computeLogReturns	computes log returns from price data
computePercentReturns	computes percent returns from price data
estimate	estimates parameters of a time series model
forecast	generates a time series and conditional variance forecast
getCV	puts conditional variances or variance-covariance matrices into global vector **_fan_CV**
getCOR	puts conditional correlations into global variable **_fan_COR**
getEstimates	puts model estimates into global variable **_fan_Estimates**
getResiduals	puts unstandardized residuals into global vector
getSeriesACF	puts autocorrelations into global variable **_fan_ACF**
getSeriesPACF	puts partial autocorrelations into global variable **_fan_PACF**
getSession	retrieves a data analysis session
getSR	puts standardized residuals into global vector
plotCOR	plots conditional correlations
plotCSD	plots conditional standard deviations
plotCV	plots conditional variances
plotQQ	generates quantile-quantile plot
plotSeries	plots time series
plotSeriesACF	plots autocorrelations
plotSeriesPACF	plots partial autocorrelations
plotSR	plots standardized residuals
session	initializes a data analysis session
setAlpha	sets inference alpha level
setConstraintType	sets type of constraints on parameters
setCovParType	sets type of covariance matrix of parameters
setCVIndEqs	declares list of independent variables to be included in conditional variance equations
setDataset	sets dataset name
setIndEqs	declares list of independent variables to be included in mean equations
setInferenceType	sets type of inference
setIndVars	declares names of independent variables

setLagTruncation	sets lags included for FIGARCH model
setLagInitialization	sets lags excluded for FIGARCH model
setLjungBoxOrder	sets order for Ljung-Box statistic
setOutputFile	sets output file name
setRange	sets range of data
setSeries	declares names of time series
setVarNames	sets variable names for data stored in ASCII file
showEstimates	displays estimates in simple format
showResults	displays results of estimations
showRuns	displays runs
simulate	generates simulation
testSR	generates skew, kurtosis, Ljung-Box statistics

2.6.1 Initializing the Session

First, an analysis session must be established.

```
session ses1 'time series analysis';
```

will start a new session, and

```
getSession ses1;
```

will retrieve a previous analysis session.

In either command, *ses1* is the name of the session and is required. It must be no more than eight characters, and the analysis results will be stored in a **GAUSS** matrix file of the same name with a *.fmt* extension. Thus the results of either of the above sessions will be stored in a file with the name *ses1.fmt*.

2.6.2 Entering Data

Before any analysis can be done, the time series must be brought into memory. If the time series resides in a **GAUSS** dataset, enter

```
setDataset stocks;
setSeries intel;
```

FANPACMT looks for a **GAUSS** dataset called stocks.dat, and then looks into that dataset for a variable with name intel. If it exists, the time series is inserted into the **FANPACMT** global **_fan_Series**.

If the time series is stored in a "flat" ASCII file, it is first necessary to declare the column names. This can be done using the **FANPACMT** keyword command, **setVarNames**:

```
setVarNames  date intel intelvol;
setDataset intel.asc;
setSeries intel;
```

The **setVarNames** command puts the variable labels into the **FANPACMT** global, **_fan_VarNames**.

2.6.3 The Date Variable

FANPACMT assumes that the first observation is the oldest and the last observation is the newest. It also assumes that the date variable, if available, is stored in the yyyymmdd format. One or the other, or both, of the conditions may not be met in an ASCII data file.

Many ASCII files containing stock data will have the date stored as mm/dd/yy or mm/dd/yyyy. **FANPACMT** will convert the dates to the standard format and the observations will be sorted. For example:

```
library fanpacmt,pgraph;
session nissan 'Analysis of Nissan daily log-returns';
setVarNames  date nsany;
setDataset nsany.asc;
setSeries nsany;
estimate run1 garch(1,3);
showResults;
```

`nsany.asc` is an ASCII file, and the command **setDataset** causes **FANPACMT** to create a **GAUSS** dataset of the data with the same name as the name of the file in the keyword command argument preceding the extension. Thus a **GAUSS** dataset with file name *nsany.dat* is created with two variables in it with variable names *date* and *nsany*. If you wish the **GAUSS** data file to have a different name, include an argument in the keyword command with the desired name of the **GAUSS** dataset. For example:

```
setDataset nsany.asc newnsany;
```

It is important that the new **GAUSS** dataset file name come after the name of the ASCII data file.

2.6.4 Scaling Data

A keyword command is available for computing log returns from price data. Thus if the time series in the dataset is price data, the log returns can be computed by entering

```
computeLogReturns 251;
```

The argument is a scale factor. This function computes

$$LR_t = \kappa \, log(\frac{P_t}{P_{t-1}})$$

where P_i is price at time i, and κ is the scale factor. For best numerical results, data should be scaled to the year time scale. Thus for monthly data, $\kappa = 12$, and for daily data, $\kappa = 251$.

An additional keyword command is available for computing log *percent* returns from price data by calling **computePercentReturns**. This function computes

$$PCTR_i = \kappa \frac{P_t - P_{t-1}}{P_{t-1}}$$

where P_i is price at time i, and κ is the scale factor. For interpretation as a percent, the scale factor should be set to 100.

```
computePercentReturns 100;
```

2.6.5 Independent Variables

To add independent variables to the session, enter their names using

```
setIndVars intelvol;
```

This command assumes that the independent variables are stored in the same location as the time series. The independent variables are stored in a **FANPACMT** global, **_fan_IndVars**.

The effect of the sequence of commands ending in setSeries is to store the time series in a global variable, **_fan_Series**; the independent variables, if any, in **_fan_IndVars**; and to store the names of the session, dataset time series and independent variables in a packed matrix on the disk.

2.6.6 Selecting Observations

A subset of the time series can be analyzed by specifying row numbers or, if a date variable exists in the dataset, by date. The date variable must be in the format, yyyymmdd. For the Intel dataset described above, the following are equivalent subsets:

```
setVarNames   date intel intelvol;
setDataset   intel.asc;
setSeries   intel 19960530 19961231;
```

or

```
setVarNames   date intel intelvol;
setDataset   intel.asc;
setSeries   intel 54 203;
```

The beginning and end of the time series may be specified by `start` and `end`:

```
setVarNames   date intel intelvol;
setDataset   intel.asc;
setSeries   intel start 19961231;
```

or

```
setVarNames   date intel intelvol;
setDataset   intel.asc;
setSeries   intel 19960530 end;
```

2.6.7 Simulation

A keyword command is available for simulating data from the various models in **FANPACMT**. First, a string array is constructed containing the information required for the simulation, and the name of this array is passed to the keyword command. For example:

```
library fanpacmt;

string ss = {

  "Model garch(1,2)",
  "NumObs 300",
  "DataSetName example",
  "TimeSeriesName Y",
  "Omega .2",
  "GarchParameter .5",
  "ArchParameter .4 -.1",
  "Constant .5",
  "Seed 7351143"
};

simulate ss;
```

This produces a simulation of a GARCH(1,2) model with 300 observations and puts it into a **GAUSS** dataset named `example`.

The following simulation parameters may be included in the string array:

Model model name (required)

NumObs number of observations (required)

DataSetName name of **GAUSS** dataset into which simulated data will be put (required)

TimeSeriesName variable label of time series

Omega GARCH process constant, required for GARCH models

GarchCoefficients GARCH coefficients, required for GARCH models

ArchCoefficients ARCH coefficients, required for GARCH models

ARCoefficients AR coefficients, required for ARIMA models

MACoefficients MA coefficients, required for ARIMA models

RegCoefficients Regression coefficients, required for OLS models

DFCoefficient degrees of freedom parameter for t-density. If set, t-density will be used; otherwise Normal density

Constant constant (required)

Seed seed for random number generator (optional)

Note: Only the first two characters of the field identifier are actually looked at.

2.6.8 Setting Type of Constraints

By Default constraints described in Nelson and Cao (1992) are imposed on GARCH(1,q) and GARCH(2,q) models to ensure stationarity and nonnegativity of conditional variances (as described in Section2.3.1). These are the least restrictive constraints for these models.

Most GARCH estimation reported in the economics literature employ more restrictive constraints for ensuring stationarity. They are invoked primarily because the optimization software does not provide for nonlinear constraints on parameters. In this case, the GARCH parameters are simultaneously constrained to be positive and to sum to less than 1. For several reasons, including comparisons with published results, you may want to impose either no constraints or the commonly employed more highly restrictive constraints. A keyword function is provided in **FANPACMT** for selecting these types of constraints:

```
setConstraintType standard
```

selects the Nelson and Cao (1992) constraints (described in Section/ref:consts). These are the least restrictive constraints that ensure stationarity and nonnegativity of the conditional variances, and are imposed by default.

```
setConstraintType unconstrained
```

will produce GARCH estimates without constraints to ensure stationarity. Nonnegativity of conditional variances is maintained by bounds constraints placed directly on the conditional variances themselves.

```
setConstraintType bounds
```

imposes the more highly restrictive linear constraints on the parameters. They constrain the coefficients in the conditional variance equation simultaneously to be greater than zero and to sum to less than one.

2.6.9 The Analysis

The **estimate** command is used for all analysis. Once the time series itself has been stored in the global, **_fan_Series**, it can be analyzed. The following performs a GARCH estimation:

```
estimate run1 garch;
estimate run2 garch(2,2);
estimate run3 egarch;
estimate run4 arima(2,1,1);
```

The first argument, the run name, is necessary. All results of this estimation will be stored in the session matrix under that name.

With the exception of OLS, these estimations are iterative using the **GAUSS SQPsolveMT** procedure. In some cases, therefore, the iterations may be time consuming. **SQPsolveMT** permits you to monitor the iterations using keystrokes. To cause **SQPsolveMT** to print iteration information to the screen, press “o”. To force termination of the iterations press “c”.

The following models may be estimated:

ols	normal linear regression model
$tols$	t distribution linear regression model
$arma(m, n)$	Normal ARMA model
$tarma(m, n)$	t distribution ARMA model
$garch(p, q)$	Normal GARCH model
$agarch(p, q)$	Normal GARCH model with asymmetry parameters
$tagarch(p, q)$	t distribution GARCH model with asymmetry
	parameters
$gtagarch(p, q)$	skew gen. t distribution GARCH model with asymmetry
	parameters
$tgarch(p, q)$	t distribution GARCH model
$gtgarch(p, q)$	skew gen. t distribution GARCH model
$igarch(p, q)$	Normal integrated GARCH model
$tigarch(p, q)$	t distribution integrated GARCH model
$gtigarch(p, q)$	skew gen. t distribution integrated GARCH model
$egarch(p, q)$	exponential GARCH model
$negarch(p, q)$	Normal GARCH model with leverage parameters
$tegarch(p, q)$	t distribution GARCH model with leverage
	parameters
$gtegarch(p, q)$	skew gen. t distribution GARCH model with leverage
	parameters
$figarch(p, q)$	Normal fractionally integrated GARCH model
$fitgarch(p, q)$	t distribution fractionally integrated GARCH model
$figtgarch(p, q)$	skew gen. t distribution fractionally integrated GARCH model
$varma(m, n)$	Normal multivariate VARMA model
$tvarma(m, n)$	t distribution multivariate VARMA model
$dvgarch(p, q)$	Normal DVEC multivariate GARCH model
$cdvgarch(p, q)$	constant correlation Normal DVEC multivariate GARCH model
$dvtgarch(p, q)$	t distribution DVEC multivariate GARCH model
$cdvtgarch(p, q)$	constant correlation t distribution DVEC multivariate GARCH model
$bkgarch(p, q)$	Normal BEKK multivariate GARCH model
$bktgarch(p, q)$	t distribution BEKK multivariate GARCH model

For an ARCH model set p = 0.

For ARMA-GARCH models use (p, q, m, n) where m is the order of the auto-regression parameters, and n the order of the moving average parameters.

2.6.10 Results

After the estimations have finished, results are printed using the command

```
showResults;
```

Results for individual runs can be printed by listing them in the command

```
showResults run1 run3;
```

2.6.11 Standardized and Unstandardized Residuals

It may be useful to generate standardized residuals and analyze their moments or plot their cumulative distribution against their predicted cumulative distributions. Thus

```
plotSR;
plotQQ;
```

produces a plot of the standardized results (for all model estimations by default, or specified ones if listed in the command), and plots the observed against the theoretical cumulative distributions. Both of these commands put the requested standardized residuals into the global **_fan_SR**. If you wish only to store the standardized residuals in the global, use

```
getSR;
```

or to get a particular standardized residual

```
getSR run2;
```

Unstandardized residuals are stored in **_fan_Residuals** with the following command

```
getResiduals run2;
```

A request can also be made to test the standardized residuals. The keyword command

```
testSR;
```

will generate an analysis of the time series and residuals. Skew and kurtosis statistics are computed and a heteroskedastic-consistent Ljung-Box statistic (Gouriéroux, 1997) is computed that tests the time series and residuals for autocorrelation. For example

```
=====================================================================
                         Session: example1
---------------------------------------------------------------------
                         wilshire example
---------------------------------------------------------------------


                   Time Series
```

```
==================================================
                  Series: cwret
--------------------------------------------------
       skew       -266.1720   pr =       0.000
   kurtosis       8558.5534   pr =       0.000

   heteroskedastic-consistent
   Ljung/Box        39.0881   pr =       0.124
--------------------------------------------------


                    Residuals

==================================================
               run1: GARCH(2,1)
--------------------------------------------------
       skew         -3.9581   pr =       0.047
   kurtosis          8.3773   pr =       0.004

   heteroskedastic-consistent
   Ljung/Box        17.2809   pr =       0.969

==================================================
               run2: TGARCH(2,1)
--------------------------------------------------
       skew         -4.3003   pr =       0.038
   kurtosis         10.4523   pr =       0.001

   heteroskedastic-consistent
   Ljung/Box        18.9296   pr =       0.941
--------------------------------------------------
```

2.6.12 Conditional Variances and Standard Deviations

For the GARCH models, the conditional variances are of particular interest. To plot these, enter

```
plotCV;
```

This also stores them in **_fan_CV**. To store them in a global without plotting, use

```
getCV;
```

In some contexts the conditional standard deviations, that is, the square roots of the conditional variances, are more useful. To generate a plot, enter

```
plotCSD;
```

If percentage scaling has been used for the time series, you may want to annualize the data by scaling. This can be done by adding a scale factor in the call to **plotCSD**. For example, if the data are monthly, enter a value of 12 for the scale factor:

```
plotCSD 12;
```

2.6.13 Example

The following example analyzes daily data on Intel common stock. Two models are fitted, the results are printed, and the conditional variances are plotted.

```
library fanpacmt,pgraph;
session wilshire 'wilshire example';
setDataset wilshire;
setSeries cwret; /* capitalization weighted returns */
setInferenceType simLimits;
estimate run1 garch(2,2);
estimate run2 garch(2,2,2,0);
showResults;
testSR;
plotCV;
```

```
===============================================================================
                              Session: wilshire
-------------------------------------------------------------------------------
                              wilshire example
-------------------------------------------------------------------------------
 FANPACMT Version 2.0.0       Data Set:  wilshire        3/05/2003 12:07:02
===============================================================================


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                  run: run1
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

return code =    0
normal convergence


Model:  GARCH
```

```
Number of Observations     : 341
Observations in likelihood : 341
Degrees of Freedom         : 335

AIC         1991.58
BIC         2014.57
LRS         1979.58

          roots
     ---------------
       1.0241991
      -1.7635304

       Abs(roots)
     ---------------
       1.0241991
       1.7635304


Maximum likelihood covariance matrix of parameters
0.95 confidence limits computed from inversion of Wald statistic

Series: cwret


  Variance Equation


Variance Equation Constant(s)

Estimate
 1.11894
standard error
 0.07007

lower confidence limit
 0.95686

upper confidence limit
 1.23437

Garch Parameter(s)

Estimate
```

```
 0.38099
 0.43180
standard error

 0.04657
 7.49337

lower confidence limit

 0.22638
-24.66137

upper confidence limit

 0.44698
 12.25291

Arch Parameter(s)

Estimate

 0.02834
 0.12185
standard error

 0.08229
 0.17828

lower confidence limit

-0.26628
-0.25567

upper confidence limit

 0.10723
 0.31078


  Mean Equations


Constant(s)

Estimate
 1.1420
```

```
standard error
 0.0511

lower confidence limit
 0.9634

upper confidence limit
 1.1420



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                   run: run2
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

return code =    0
normal convergence


Model:  GARCH

Number of Observations     : 341
Observations in likelihood : 341
Degrees of Freedom         : 333

AIC         1994.36
BIC         2025.02
LRS         1978.36

          roots
     ---------------
       1.0274493
      -1.8113846
      0.25899372 +        4.2630194i
      0.25899372 -        4.2630194i

       Abs(roots)
     ---------------
       1.0274493
       1.8113846
       4.2708796
       4.2708796
```

FANPACMT

```
Maximum likelihood covariance matrix of parameters
0.95 confidence limits computed from inversion of Wald statistic

Series: cwret


  Variance Equation


Variance Equation Constant(s)

Estimate
 1.11880
standard error
 0.05325

lower confidence limit
 0.97001

upper confidence limit
 1.16972

Garch Parameter(s)

Estimate

 0.38530
 0.44236
standard error

 0.05406
 10.72465

lower confidence limit

 0.18941
-39.06050

upper confidence limit

 0.39583
 5.74083

Arch Parameter(s)

Estimate
```

```
 0.03592
 0.09496
standard error

 0.10721
 0.21764

lower confidence limit

-0.36311
-0.65643

upper confidence limit

 0.12166
 0.21019


  Mean Equations


Constant(s)

Estimate
 1.1793
standard error
 180.5153

lower confidence limit
-608.9496

upper confidence limit
 1.1793

AR Parameter(s)

Estimate

 0.0284
-0.0548
standard error

 0.0198
 0.0211
```

FANPACMT

```
lower confidence limit

-0.0248
-0.1295

upper confidence limit

 0.0485
-0.0548


===============================================================================
                              Session: wilshire
-------------------------------------------------------------------------------
                              wilshire example
-------------------------------------------------------------------------------

                  Time Series

==================================================
                 Series: cwret
--------------------------------------------------
       skew      -360.8689   pr =      0.000
   kurtosis      9266.8838   pr =      0.000

   heteroskedastic-consistent
   Ljung-Box       42.4523   pr =      0.065
--------------------------------------------------


                   Residuals

==================================================
                run1: GARCH(2,2)
--------------------------------------------------
       skew      -357.0129   pr =      0.000
   kurtosis      9280.9146   pr =      0.000

   heteroskedastic-consistent
   Ljung-Box       21.1015   pr =      0.885

==================================================
            run2: GARCH(2,2,2,0)
--------------------------------------------------
       skew      -335.4611   pr =      0.000
```

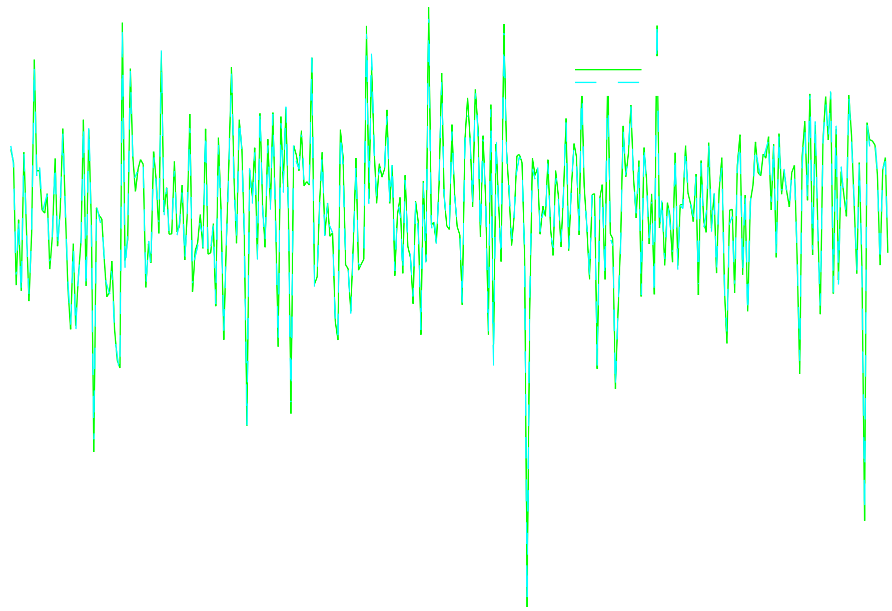

Figure 2.1: Plot of conditional variances for ARCH and GARCH models on a 27-year monthly series of the capitalization-weighted Wilshire 5000 index

```
   kurtosis      9034.2040   pr =      0.000

   heteroskedastic-consistent
   Ljung-Box       20.7854   pr =      0.894
--------------------------------------------------
```

2.6.14 Altering SQPSolveMT control variables

When an estimation is invoked (i.e., when **estimate** is called) **FANPACMT** a default **SQPsolveMT** control structure is constructed. If you wish to alter any of the SQPsolveMT control variables add a proc to the command file that takes an SQPsolveMTControl structure as an input argument and output argument. In the proc modify the control variables as desired.

```
proc sqpcont( struct sqpsolvemtcontrol c0 );
    c0.maxiters = 100;
    c0.printiters = 1;
    retp(c0);
endp;

struct fanControl f0;
f0 = fanControlCreate;
f0.sqpsolvemtControlProc = &sqpcont;
```

Be aware that some of the control variables are required for the estimation and their modification may produce unpredictable results. If you wish, for example, to add linear constraints to the model, you must test first whether the linear constraint matrices have already been set:

```
proc sqpcont( struct sqpsolvemtcontrol c0 );

    if rows(c0.A) == 0;
        c0.A = 1~0~0~0~0~-1;
        c0.b = 1;
    else;
        c0.A = c0.A | (1~0~0~0~0~-1);
        c0.b = c0.B | 1;
    endif;

    retp(c0);
endp;

struct fanControl f0;
f0 = fanControlCreate;
f0.sqpsolvemtControlProc = &sqpcont;
```

2.6.15 Multivariate Models

Most keyword commands behave in the same way for multivariate models as for univariate. The specification of the time series being analyzed, for example, merely requires adding another name to the keyword command

```
setSeries AMZN YHOO;
```

The specification of the independent variables is slightly different. **FANPACMT** allows specifying different sets of independent variables for each equation. A simple list of independent variables, as is done for the unvariate models, causes all independent variables to be included in all equations:

```
setIndVars AMZNvol YHOOvol
```

To specify a different list of independent variables for each equation, add the name of the dependent variable to the list, and call **setIndVars** for each dependent variable as needed. Any equation for which **setIndVars** is not called will contain all the independent variables.

```
setIndVars AMZN AMZNvol
setIndVars YHOO YHOOvol;
```


2.6.16 Example

```
library fanpacmt,pgraph;

session mult 'May 15, 1997 to November 9, 1998';

setDataSet stocks;
setSeries AMZN YHOO;
setIndVars *YHOOvol *AMZNvol;
setIndEqs AMZN lnAMZNvol;
setIndEqs YHOO lnYHOOvol;
setCVIndEqs AMZN lnAMZNvol;
setCVIndEqs YHOO lnYHOOvol;
setSqrtCV on;
setInferenceType simBound;
setStationarityConstraint roots;

computeLogReturns 251;

estimate run1 cdvgarchv(2,1);
showResults;
plotCV;
```

```
===============================================================================
                                 Session: mult
-------------------------------------------------------------------------------
                       May 15, 1997 to November 9, 1998
-------------------------------------------------------------------------------
 FANPACMT Version 2.0.0          Data Set:  stocks         3/06/2003 13:33:54
===============================================================================


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
                                 run: run1
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

return code =    0
normal convergence


Model:  CDVGARCHV

Number of Observations     : 375
Observations in likelihood : 375
Degrees of Freedom         : 360

AIC         4505.97
BIC         4564.87
LRS         4475.97

          roots
     ---------------
       9.4281112
               1
       46.223078
       1.0000341
       3.5668614
       3.5668614
       27.960257
        2.261289

       Abs(roots)
     ---------------
       9.4281112
               1
       46.223078
       1.0000341
       3.5668614
       3.5668614
       27.960257
        2.261289


Maximum likelihood covariance matrix of parameters
0.95 confidence limits computed from inversion of Wald statistic

Series 1: AMZN
Series 2: YHOO
```

```
  Variance Equation


Variance Equation Constant(s)

Estimate

 3.92334
 2.55377
standard error

 19.82638
 21.41505

lower confidence limit

-81.88559
-81.84615

upper confidence limit

 3.92334
 2.55377

Garch Parameter(s)

Estimate

 0.37878    0.34116
-0.04795   -0.01059
standard error

 0.04611    0.18736
 0.06490    0.25303

lower confidence limit

 0.17216   -0.53310
-0.30663   -1.16765

upper confidence limit

 0.37878    0.34116
-0.04795   -0.01059
```

```
Arch Parameter(s)

Estimate
 0.21711    0.18001
standard error
 0.03088    0.02506

lower confidence limit
 0.05836    0.05761

upper confidence limit
 0.21711    0.18001

Variance Equation Regression Coefficient(s)

Estimate

 0.00000    0.59439
 0.57870    0.00000
standard error

 0.00000    0.01721
 0.01530    0.00000

lower confidence limit

 0.00000    0.52353
 0.51587    0.00000

upper confidence limit

 0.00000    0.59439
 0.57870    0.00000


  Mean Equations


Constant(s)

Estimate

-6.2382
-6.5485
standard error
```

```
 0.0545
 0.0621

lower confidence limit

-6.4206
-6.6672

upper confidence limit

-6.2382
-6.4871

Regression coefficient(s)

Estimate

 0.0000       0.7642
 0.7222       0.0000
standard error

 0.0000       0.4687
 0.6427       0.0000

lower confidence limit

 0.0000      -0.9480
-1.3597       0.0000

upper confidence limit

 0.0000       0.7642
 1.3099       0.0000


  Miscellaneous Parameters


VC

Estimate

 1.0000       0.4433
 0.4433       1.0000
```

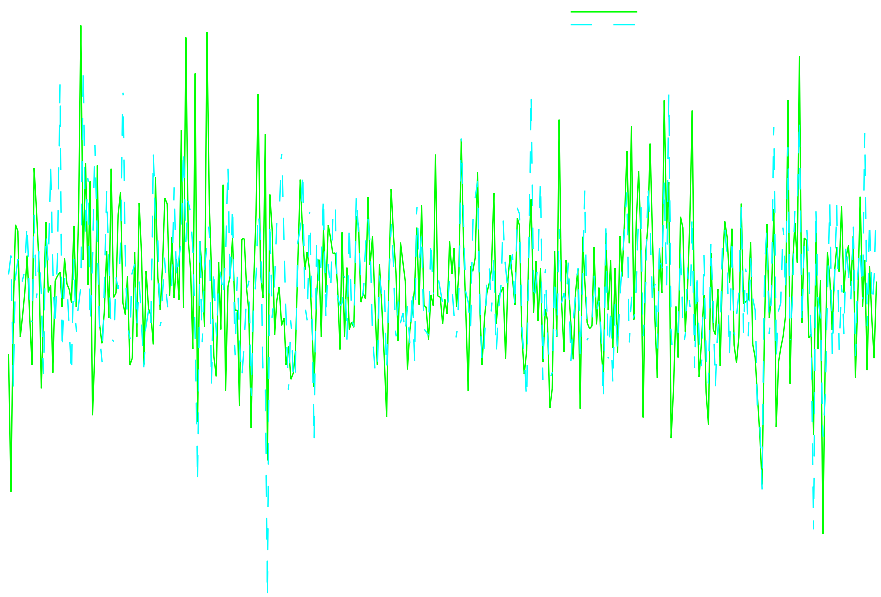

Figure 2.2: Plot of conditional variances for AMZN and YHOO using a Constant Correlation Diagonal Vec multivariate GARCH model

```
standard error

 1.0000        0.0147
 0.0147        1.0000

lower confidence limit

 1.0000        0.4416
 0.4416        1.0000

upper confidence limit

 1.0000        0.5055
 0.5055        1.0000
```

2.7 Data Transformations

Box-Cox

This transformation involves estimating additional parameters. The Box-Cox transformation is

$$f(x) = \frac{x^\lambda - 1}{\lambda}$$

where λ is an estimated parameter.

When using the **ugarch** or **mgarch** procedures for estimation, this transformation is specified by setting the **bxcx** member of the **fanControl** structure for independent variables, or the **seriesbxcx** member for the time series variables.

The keyword command **setBoxcox** can be used to set this transformation for selected variables in the model.

Computing Returns

Two keyword commands are available for computing returns from prices in time series. **computeLogReturns** computes

$$R_i = Kln(P_i/P_{i-1});$$

where K is a scale factor. The scale factor is specified as an argument in the keyword call:

```
computeLogReturns 251;
```

The keyword command **computePercentReturns** computes

$$R_i = K(P_i - P_{i-1})/P_{i-1}$$

where K is a scale factor. For a time series with interpretation as a proportion $K = 1$, or as a percent, $K = 100$.

Log transformations on Independent Variables

The keyword command **setIndVars** can be used to specify a log transformation of selected independent variables. Insert an asterisk in front of the selected variables in the variable list. Then in later references to those variables in keyword commands, append **ln** to its name. For an illustration see the example in Section2.6.15.

Creating Elapsed Time between Observations

Two keywords will generate independent variables measuring the elapsed time between observations in the time series, **setIndLogElapsedPeriod** and **setIndElapsedPeriod**. The variable names for the generated variablers are **lnEP** and **EP** respectively.

```
library fanpacmt,pgraph;
session wilshire 'wilshire example';
setDataset wilshire;
setSeries cwret; /* capitalization weighted returns */
setLogElapsedPeriod;
setIndEqs lnEP;

estimate run1 garch(2,2);

showResults;
```

2.8 FANPACMT Session Structure

When using the keywords **FANPACMT** saves results and various aspects of the problem in a **fanSession** structure. The following is the definition of that structure (from `fanpacmt.sdf`):

```
struct fanSession {
          string name;
          string title;
          string array runNames;
          string dataset;
          matrix series;
          matrix indvars;
          matrix range;
          matrix scale;
          matrix version;
          string date;
          matrix stddev;
          matrix mean;
          matrix mnind;
          matrix stdind;
          struct fanEstimation Runs;
             };
```

The instance of a **fanSession** structure is saved to the disk with the the name given in the **session** keyword command as session name. The member `runNames` contains the names of all of the runs in that session given in the **estimation** keyword command. Associated with each of those run names is an element of the **fanEstimation** instance with the results of those runs.

The **fanEstimation** structure has the following definition:

```
struct fanEstimation {

           string name;
           scalar model;
           string title;
           struct fanControl control;
           scalar aic;
           scalar bic;
           scalar lrs;
           scalar numObs;
           scalar df;

           struct PV par;
           scalar retcode;
           matrix moment;
           matrix hessian;
           matrix climits;
           matrix tsforecast;
           matrix cvforecast;

                 };
```

A **PV** structure is nested in this structure containing the parameter estimates. The **PV** structure doesn't have any public members but rather there are a set of functions associated with the **PV** structure. Two of these are used to retrieve the parameters. **pvUnpack** retrieves them in their original form as matrices or arrays. **pvGetParvector** retrieves the entire vector of estimated parameters.

The parameter names stored in **par**, the instance of the **PV** structure in **fanSession** are

1 **beta0**, constants in mean equations

2 **beta**, regression coefficients

3 **lambda_y**, Box-Cox coefficients

4 **lambda_x**, Box-Cox coefficients

5 **omega**, constants in conditional variance equations

6 **garch**, garch coefficients

7 **arch**, arch coefficients

8 **ar**, auto-regression coefficients

9 **ma**, moving average coefficients

10 **gamma**, inCV coefficients

11 **nu**, “degrees of freedom” parameter for t-distribution

12 **rho**, shape parameter for exponential distribution

13 **dm**, fractional integration parameter

14 **zeta**, leverage parameter for egarch model

15 **delta**, in mean coefficient for conditional variance

16 **delta_s**, in mean coefficient conditional standard deviation

17 **s2**, covariance matrix for multivariate t

18 **tau**, assymetry parameter

The parameters may be unpacked using either the number or the name:

```
garch = pvUnpack(f0.runs[1].par,"garch");
arch = pvUnpack(f0.runs[1].par,7);
```

Example

```
>> run fanpacmt.sdf
>> struct fanSession f0
>> { f0, ret } = loadStruct("wilshire","fanSession");
>> print f0.runnames

       run1
       run2

>> print pvUnpack(f0.runs[1].par,"garch")

      0.3810
      0.4318

>> print pvUnpack(f0.runs[2].par,"garch")
```

```
    0.3853
    0.4424


>> print pvGetParVector(f0.runs[1].par);

   1.1420
   1.1189
   0.3810
   0.4318
   0.0283
   0.1218

>> print f0.runs[1].lrs

   1979.5753

>>  f0.runs[2].lrs;

   1978.3599
```

2.9 FANPACMT Procedures

The **FANPACMT** procedures used by the keyword commands can be called directly. The maximum likelihood procedures for each of the **FANPACMT** models can be put into command files and estimates generated using the **SQPsolveMT** optimization procedures.

For example, the following is a command file for estimating a GARCH model. It estimates the model in two ways: first, using the Nelson and Cao constraints; and second, using standard constraints. The results follow the command file.

```
library fanpacmt;
#include fanpacmt.sdf

Y = loadd("example");

struct DS d0;
d0.dataMatrix = Y;

struct fanControl c0;
c0 = fanControlCreate;

c0.p = 3;
```

```
c0.q = 2;

c0.cvConstType = 1;  /* Nelson and Cao constraints  */

struct fanEstimation f0;
f0 = ugarch(c0,d0);

print;
print;
print "       Nelson & Cao constraints";
print;

lbl = pvGetParnames(f0.par);
p = pvGetParVector(f0.par);

format /rd 12,4;
print "        Coefficients         lower cl    upper cl";
for i(1,rows(p),1);
    print lbl[i];;
    print p[i];;
    print f0.climits[i,.];
endfor;


    Nelson & Cao constraints

       Coefficients         lower cl    upper cl
 beta0[1,1]      0.4568       0.3796      0.5340
 omega[1,1]      0.4129       0.0229      0.8030
 garch[1,1]     -0.5595      -1.2238      0.1049
 garch[2,1]      0.1137      -0.0545      0.2820
 garch[3,1]      0.0647      -0.1385      0.2678
 arch[1,1]       0.4904       0.2653      0.7155
 arch[2,1]       0.4713       0.1297      0.8129
```

2.9.1 Bibliography

Amemiya, Takeshi, 1985. *Advanced Econometrics.* Cambridge, MA: Harvard University Press.

Baillie, Richard T., Bollerslev, Tim, and Mikkelsen, Hans Ole Æ., 1996. "Fractionally integrated generalized autoregressive conditional heteroskedasticity", *Journal of Econometrics*, 74:3-28.

Gallant, A. R., 1987. *Nonlinear Statistical Models.* New York: Wiley.

Geweke, John, 1995. “Posterior simulators in econometrics,” Working Paper 555, Research Department, Federal Reserve Bank of Minneapolis.

Glosten, L.R., Jagannathan, R., and Runkle, D.E., 1993. “On the relation between the expected value and the volatility of the nominal excess return on stocks”, *Journal of Finance*, 48(5):1779-1801.

Gouriéroux, Christian, 1997. *ARCH Models and Financial Applications.* New York: Springer-Verlag.

Gouriéroux, Christian, Holly, Alberto, and Monfort, Alain, 1982. “Likelihood ratio test, Wald Test, and Kuhn-Tucker test in linear models with inequality constraints on the regression parameters,” *Econometrica*, 50:63-80.

Hartmann, Wolfgang M. and Hartwig, Robert E., 1995. “Computing the Penrose-Moore inverse for the covariance matrix in constrained nonlinear estimation,” SAS Institute, Inc., Cary, NC.

Nelson, Daniel B., 1991. “Conditional Heteroskedasticity in Asset Returns: a New Approach”, *Econometrica*, 59(2):347-370.

Nelson, Daniel B. and Cao, Charles Q., 1992. “Inequality constraints in the univariate GARCH model,” *Journal of Business and Economic Statistics*, 10:229-235.

O’Leary, Dianne P. and Rust, Bert W., 1986. “Confidence intervals for inequality-constrained least squares problems, with applications to ill-posed problems,” *American Journal for Scientific and Statistical Computing*, 7(2):473-489.

Schoenberg, Ronald J., 1997. “Constrained maximum likelihood,” *Computational Economics*, 10:251-266.

Self, Steven G. and Liang, Kung-Yee, 1987. “Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions,” *Journal of the American Statistical Association*, 82:605-610.

Theodossiou, Panayiotis, 1998. “Financial data and the skewed generalized t distribution”, *Management Science*, 44:1650-1661.

White, H., 1981. “Consequences and detection of misspecified nonlinear regression models.” *Journal of the American Statistical Association*, 76:419-433.

White, H., 1982. “Maximum likelihood estimation of misspecified models,” *Econometrica*, 50:1-25.

Wolak, Frank, 1991. “The local nature of hypothesis tests involving inequality constraints in nonlinear models,” *Econometrica*, 59:981-995.

Chapter 3

FANPACMT Keyword Reference

Summary of Keyword Commands

clearSession	clears session from memory, resets global variables
constrainPDCovPar	sets **NLP** global for constraining covariance matrix of parameters to be positive definite
computeReturns	computes returns from price data
computeLogReturns	computes log returns from price data
computePercentReturns	computes percent returns from price data
estimate	estimates parameters of a time series model
forecast	generates a time series and conditional variance forecast
getCV	puts conditional variances or variance-covariance matrices into global vector **_fan_CV**
getCOR	puts conditional correlations into global variable **_fan_COR**
getEstimates	puts model estimates into global variable **_fan_Estimates**
getResiduals	puts unstandardized residuals into global vector
getSeriesACF	puts autocorrelations into global variable **_fan_ACF**
getSeriesPACF	puts partial autocorrelations into global variable **_fan_PACF**
getSession	retrieves a data analysis session
getSR	puts standardized residuals into global vector
plotCOR	plots conditional correlations
plotCSD	plots conditional standard deviations
plotCV	plots conditional variances
plotQQ	generates quantile-quantile plot
plotSeries	plots time series
plotSeriesACF	plots autocorrelations
plotSeriesPACF	plots partial autocorrelations
plotSR	plots standardized residuals
session	initializes a data analysis session
setAlpha	sets inference alpha level
setBoxcox	indicates variables for Box-Cox transformation
setConstraintType	sets type of constraints on parameters
setCovParType	sets type of covariance matrix of parameters

setCVIndEqs	declares list of independent variables to be included in conditional variance equations
setDataset	sets dataset name
setIndElapsedPeriod	creates independent variable measuring elapsed time between observations
setIndEqs	declares list of independent variables for a particular run
setIndLogElapsedPeriod	creates independent variable measuring elapsed time between observations
setInferenceType	sets type of inference
setIndVars	establishes list of independent variables for session
setInmean	sets inMean model
setLagTruncation	sets lags included for FIGARCH model
setLagInitialization	sets lags excluded for FIGARCH model
setLjungBoxOrder	sets order for Ljung-Box statistic
setOutputFile	sets output file name
setRange	sets range of data
setSeries	declares names of time series
setVarNames	sets variable names for data stored in ASCII file
showEstimates	displays estimates in simple format
showResults	displays results of estimations
showRuns	displays runs
simulate	generates simulation
testSR	generates skew, kurtosis, Ljung-Box statistics

■ Purpose

Resets globals to default values.

■ Library

`fanpacmt`

■ Format

clearSession;

■ Source

`fankeymt.src`

Purpose

Sets **NLP** global for constraining covariance matrix of parameters to be positive definite

Library

`fanpacmt`

Format

constrainPDCovPar [*action*];

Input

action String. If absent, constraint feature is turned off, otherwise, set to

ON feature is turned on,

OFF feature is turned off,

Global Output

_gg_constPDCovPar Scalar, internal **FANPACMT** global. If nonzero, the **NLP** global **_nlp_ConstrainHess** is set to a nonzero value, causing **NLP** to construct equality constraints to handle linear dependencies in the Hessian.

Remarks

If an equality constraint is so constructed by **NLP** at convergence, it will be used in calculating the covariance matrix of the parameters. This equality constraint is stored by **NLP** in the **NLP** global, **_nlp_PDA** and is reported by the **FANPACMT** keyword command **showResults**.

Source

`fankeymt.src`

Purpose

Computes log returns from price data.

Library

`fanpacmt`

Format

computeLogReturns [*list*] [*scale*];

Input

list	List of time series. Default, all time series.
scale	Scale factor. If omitted, scale factor is set to one.

Global Input

_fan_Series	N×k matrix, time series.

Global Output

_fan_Series	N×k matrix, time series.

Remarks

Computes the log returns from price data.

$$R_i = \kappa \, log\left(\frac{P_i}{P_{i-1}}\right)$$

where P_i is the price at time i and κ is the scale factor. For best numerical results, use a scale factor that scales the time units of the series to a year. Thus for monthly data, $\sigma = 12$, and for daily data, $\sigma = 251$.

Source

`fankeymt.src`

Purpose

Computes percent returns from price data.

Library

`fanpacmt`

Format

computePercentReturns [*list*] [*scale*];

Input

list	List of time series. Default, all time series.
scale	Scale factor. If omitted, scale factor is set to 100.

Global Input

_fan_Series	N×k matrix, time series.

Global Output

_fan_Series	N×k matrix, time series.

Remarks

Computes the percent returns from price data.

$$R_i = \kappa \left(\frac{P_i - P_{i-1}}{P_{i-1}} \right)$$

where P_i is the price at time i and κ is the scale factor. For interpretation as a "percent," use the default scale factor of 100.

Source

`fankeymt.src`

Purpose

Generates estimates of parameters of a time series model.

Library

`fanpacmt`

Format

estimate *run_name* **[***run_title***]** *model***;**

Input

run_name Name of estimation run. It must come first and it cannot contain embedded blanks.

run_title Title of run, put in SINGLE quotes if title contains embedded blanks. May be omitted.

model Type of time series model:

If a GARCH model name is appended with an M, an inmean model is estimated, and if with a V, an inCV model is estimated.

OLS Normal ordinary least squares.

TOLS t distribution ordinary least squares.

ARIMA(r,d,s) Normal ARIMA. If r, d, and s are not specified, an ARIMA(1,1,1) is estimated.

TARIMA(p,d,q) t distribution ARIMA.

EGARCH EGARCH with generalized error distribution.

NEGARCH EGARCH with Normal distribution

TEGARCH EGARCH with t distribution

GTEGARCH EGARCH with skew generalized t distribution

AGARCH(p,q,r,s) Normal GARCH with asymmetry parameters.

TAGARCH(p,q,r,s) Student's t distribution GARCH with asymmetry parameters.

GTAGARCH(p,q,r,s) Skew Generalized t distribution GARCH with asymmetry parameters.

GARCH(p,q,r,s) Normal GARCH.

TGARCH(p,q,r,s) Student's t distribution GARCH.

GTGARCH(p,q,r,s) Skew Generalized t distribution GARCH.

IGARCH(p,q,r,s) Normal integrated GARCH.

ITGARCH(p,q,r,s) t distribution integrated GARCH.

IGTGARCH(p,q,r,s) Skew Generalized t distribution integrated GARCH.

FIGARCH(p,q,r,s) Normal fractionally integrated GARCH.

FITGARCH(p,q,r,s) t distribution fractionally integrated GARCH.

FIGTGARCH(p,q,r,s) skew generalized t distribution fractionally integrated GARCH.

DVGARCH(p,q,r,s) Normal diagonal vec multivariate GARCH.

DVTGARCH(p,q,r,s) t distribution diagonal vec multivariate GARCH.

CDVGARCH(p,q,r,s) Normal constant correlation diagonal vec multivariate GARCH.

CDVTGARCH(p,q,r,s) t distribution constant correlation diagonal vec multivariate GARCH.

BKGARCH(p,q,r,s) Normal BEKK multivariate GARCH.

BKTGARCH(p,q,r,s) t distribution BEKK multivariate GARCH.

VARMA(r,s) Normal VARMA

TVARMA(r,s) t distribution VARMA

Global Input

_fan_Dataset Name of **GAUSS** data set containing time series being analyzed.

_fan_SeriesNames Name of time series being analyzed.

_fan_IndVarNames K×1, character vector of labels of independent variables.

Remarks

estimate generates estimates of the parameters of the specified model. The results are stored in a **GAUSS** .fmt file on the disk in the form of a vpacked matrix. These results are not printed by estimate. See **showResults** for displaying results.

All models except OLS are estimated using the **sqpSolveMT** optimization program. See the **sqpSolveMT** procedure in the **GAUSS** Run-Time Library documentation for details concerning the optimization.

Example

```
library fanpacmt,pgraph;

session test 'test session';

setDataset stocks;
setSeries intel;
setOutputfile test.out reset;

estimate run1 garch;
estimate run2 garch(2,1);
estimate run3 arima(1,2,1);

showResults;
plotSeries;
plotCV;
```

■ Source

fankeymt.src

Purpose

Generates forecasts of a time series model.

Library

`fanpacmt`

Format

forecast [*list*] [*periods*];

Input

list	Names of run for forecast. If none is specified, forecasts will be generated for all runs.
periods	Number of periods to be forecast. If not specified, the forecast is for one period.

Global Output

_fan_TSforecast	L×K matrix, L forecasts for K models.
_fan_CVforecast	L×K matrix, L forecasts for K models.

Remarks

If the model is a GARCH model, a forecast of the conditional variance is generated as well. The forecasts are written to a **FANPACMT** global. The time series forecast is written to **_fan_TSforecast** and the conditional variance is written to **_fan_CVforecast**. If **plotCV** or **plotCSD** is called after the call to **forecast**, the forecasts are included in the plot. If **plotSeries** is called after the call to **forecast**, the time series forecast is plotted with the time series as well.

Source

`fankeymt.src`

■ Purpose

Computes conditional variances and puts them into a global variable.

■ Library

`fanpacmt`

■ Format

getCV [*list*];

■ Input

list	List of runs. If omitted, conditional variances will be produced for all runs.

■ Global Output

_fan_CV	N×K matrix, conditional variances.

■ Remarks

Conditional variances are relevant only for ARCH/GARCH models. No results are generated for other models.

■ See also

plotCV

■ Source

`fankeymt.src`

Purpose

Computes conditional correlations and puts them into a global variable.

Library

`fanpacmt`

Format

getCOR [*list*];

Input

list	List of runs. If omitted, conditional correlations will be produced for all runs.

Global Output

_fan_COR	N×K matrix, conditional correlations

Remarks

Conditional correlations are relevant only for multivariate ARCH/GARCH models. No results are generated for other models.

See also

plotCOR

Source

`fankeymt.src`

■ Purpose

Stores estimates in global variable.

■ Library

`fanpacmt`

■ Format

getEstimates [*list***];**

■ Input

list	List of runs. If omitted, estimates for all runs will be stored in global variable.

■ Global Output

_fan_Estimates	K×L matrix, global into which estimates are stored.

■ Remarks

■ Source

`fankeymt.src`

Purpose

Computes unstandardized residuals and puts them into a global variable.

Library

`fanpacmt`

Format

getRD [*list*];

Input

list	List of runs. If omitted, standardized residuals will be produced for all runs.

Global Output

_fan_Residuals	N×K matrix, standardized residuals.

Source

`fankeymt.src`

Purpose

Computes autocorrelation function and puts the vector into a global variable.

Library

`fanpacmt`

Format

getSeriesACF [*list***]** *num diff* **;**

Input

list	List of series. If omitted, will be produced for all series.
num	Scalar, maximum number of autocorrelations to compute. If omitted, set to number of observations.
diff	Scalar, order of differencing. If omitted, set to zero.

Global Output

_fan_ACF	*num*×K matrix, autocorrelations.

Remarks

If one number is entered as an argument, *num* will be set to that value. If two numbers are entered as arguments, *num* will be set to the larger number and *diff* to the smaller number.

See also

plotSeriesACF, **plotSeriesPACF**, **getSeriesPACF**

Source

`fankeymt.src`

Purpose

Computes autocorrelation function and puts the vector into a global variable.

Library

`fanpacmt`

Format

getSeriesPACF [*list***]** *num diff* **;**

Input

list	List of series. If omitted, will be produced for all series.
num	Scalar, maximum number of autocorrelations to compute. If omitted, set to number of observations.
diff	Scalar, order of differencing. If omitted, set to zero.

Global Output

_fan_PACF	*num*×K matrix, autocorrelations.

Remarks

If one number is entered as an argument, *num* will be set to that value. If two numbers are entered as arguments, *num* will be set to the larger number and *diff* to the smaller number.

See also

plotSeriesPACF, **plotSeriesACF**, **getSeriesACF**

Source

`fankeymt.src`

Keyword Reference

Purpose

Retrieves a data analysis session.

Library

`fanpacmt`

Format

getSession *session_name*;

Input

session_name Name of an existing session.

Remarks

getSession retrieves a session created by a previous analysis.

Source

`fankeymt.src`

Purpose

Computes standardized residuals and puts them into a global variable.

Library

`fanpacmt`

Format

getSR [*list*];

Input

list	List of runs. If omitted, standardized residuals will be produced for all runs.

Global Output

_fan_SR	N×K matrix, standardized residuals.

See also

plotSR

Source

`fankeymt.src`

■ Purpose

Plots conditional correlations.

■ Library

`fanpacmt, pgraph`

■ Format

plotCOR [*list***] [***start end***];**

■ Input

list	List of runs. If no list, conditional correlations will be plotted for all runs.
start	Scalar, starting row or date to be included in plot. If row number, it must be greater than 1 and less than *end*. If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date." Setting *start* to START is equivalent to first observation.
end	Scalar, ending row or date to be included in plot. If row number, it must be greater than *start* and less than or equal to the number of observations. If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date." Setting *end* to END is equivalent to last observation.

■ Global Output

_fan_COR	N×K matrix, conditional correlations.

■ Remarks

Conditional correlations are relevant only for multivariate ARCH/GARCH models. No plots or output are generated for other models.

■ Source

`fanplotmt.src`

Purpose

Plots conditional standard deviations.

Library

`fanpacmt, pgraph`

Format

plotCSD [*list***] [***start end***] [***scale***];**

Input

list List of runs. If no list, conditional variances will be plotted for all runs.

start Scalar, starting row or date to be included in plot. If row number, it must be greater than 1 and less than *end*.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *start* to START is equivalent to first observation.

end Scalar, ending row or date to be included in plot. If row number, it must be greater than *start* and less than or equal to the number of observations.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *end* to END is equivalent to last observation.

scale Scalar, scale factor. The conditional standard deviations are multiplied by the square root of the scale factor before plotting. Default = 1.

Global Input

_fan_CVforecast L×K matrix, forecasts of conditional variances.

Global Output

_fan_CV N×K matrix, conditional variances.

Remarks

Conditional standard deviations are relevant only for ARCH/GARCH models. No plots or output are generated for other models.

plotCSD plots the square roots of the conditional variances times the scale factor, if any.

If **plotCSD** is called after a call to **forecast**, the square root of the forecasts of the conditional variances stored in **_fan_CVforecast** are plotted as well.

Source

`fanplotmt.src`

Purpose

Plots conditional variances.

Library

`fanpacmt, pgraph`

Format

plotCV [*list***] [***start end***];**

Input

list List of runs. If no list, conditional variances will be plotted for all runs.

start Scalar, starting row or date to be included in plot. If row number, it must be greater than 1 and less than *end*.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *start* to START is equivalent to first observation.

end Scalar, ending row or date to be included in plot. If row number, it must be greater than *start* and less than or equal to the number of observations.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *end* to END is equivalent to last observation.

Global Output

_fan_CV N×K matrix, conditional variances.

_fan_CVforecast L×K matrix, forecasts of conditional variances.

Remarks

Conditional variances are relevant only for ARCH/GARCH models. No plots or output are generated for other models.

If **plotCV** is called after a call to **forecast**, the forecasts of the conditional variance stored in **_fan_CVforecast** are plotted as well.

Source

`fanplotmt.src`

Purpose

Plots quantile-quantile plot.

Library

`fanpacmt, pgraph`

Format

plotQQ [*list*];

Input

list	List of runs. If no list, QQ plots will be generated for all runs.

Global Output

_fan_SR	N×K matrix, standardized residuals.

Source

`fanplotmt.src`

Purpose

Plots time series.

Library

`fanpacmt, pgraph`

Format

plotSeries [*list***] [***start end***];**

Input

list List of series. If no list, all series will be plotted.

start Scalar, starting row or date to be included in plot. If row number, it must be greater than 1 and less than *end*.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *start* to START is equivalent to first observation.

end Scalar, ending row or date to be included in plot. If row number, it must be greater than *start* and less than or equal to the number of observations.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *end* to END is equivalent to last observation.

Global Input

_fan_Series N×1 vector, time series.

_fan_TSforecast L×1 vector, forecasts.

Remarks

If **forecast** is called before **plotSeries**, the time series forecast stored in **_fan_TSforecast** is included in the plot.

Source

`fanplotmt.src`

Purpose

Computes autocorrelation function and puts the vector into a global variable.

Library

`fanpacmt, pgraph`

Format

plotSeriesACF [*list*] [*num*] [*diff*];

Input

list	List of series. If omitted, will be produced for all series.
num	Scalar, maximum number of autocorrelations to compute. If omitted, set to number of observations.
diff	Scalar, order of differencing. If omitted, set to zero.

Global Output

_fan_ACF	*num*×K matrix, autocorrelations.

Remarks

If one number is entered as an argument, *num* will be set to that value. If two numbers are entered as arguments, *num* will be set to the larger number and *diff* to the smaller number.

See also

plotSeriesPACF, **getSeriesACF**, **getSeriesPACF**

Source

`fanplotmt.src`

Purpose

Computes autocorrelation function and puts the vector into a global variable.

Library

`fanpacmt, pgraph`

Format

plotSeriesPACF [*list***] [***num***] [***diff***];**

Input

list	List of series. If omitted, will be produced for all series.
num	Scalar, maximum number of autocorrelations to compute. If omitted, set to number of observations.
diff	Scalar, order of differencing. If omitted, set to zero.

Global Output

_fan_PACF	*num*×K matrix, autocorrelations.

Remarks

If one number is entered as an argument, *num* will be set to that value. If two numbers are entered as arguments, *num* will be set to the larger number and *diff* to the smaller number.

See also

plotSeriesACF, **getSeriesPACF**, **getSeriesACF**

Source

`fanplotmt.src`

Purpose

Plots standardized residuals.

Library

`fanpacmt, pgraph`

Format

plotSR [*list*] [*start end*];

Input

list	List of runs. If no list, standardized residuals will be plotted for all runs.
start	Scalar, starting row or date to be included in plot. If row number, it must be greater than 1 and less than *end*. If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date." Setting *start* to START is equivalent to first observation.
end	Scalar, ending row or date to be included in plot. If row number, it must be greater than *start* and less than or equal to the number of observations. If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date." Setting *end* to END is equivalent to last observation.

Global Output

_fan_SR	N×K matrix, standardized residuals.

Remarks

Standardized residuals are relevant only for ARCH/GARCH models. No plots or output are generated for other models.

Source

`fanplotmt.src`

Purpose

Initializes a data analysis session.

Library

`fanpacmt`

Format

session *session_name* **[***session_title***];**

Input

session_name Name of session; it must contain no more than 8 characters and no embeddded blanks.

session_title Title of run, put in SINGLE quotes if title contains embedded blanks. If no title entered, it is set to null string.

Source

`fankeymt.src`

■ Purpose

Sets confidence level for statistical inference.

■ Library

`fanpacmt`

■ Format

setAlpha *alpha*;

■ Input

alpha	Scalar, confidence level. Default = .05.

■ Source

`fankeymt.src`

Purpose

sets type of constraints for stationarity conditions in Garch models

Library

`fanpacmt`

Format

SetConstraintType *type*;

Input

type String, type of constraint

standard standard constraints
bounds bounds constraints on parameters
unconstrained no constraints

Global Output

_gg_ConstType Scalar, type of constraints

1 standard constraints
2 bounds constraints on parameters
3 no constraints

Remarks

standard For garch(1,q) and garch(2,q) models, parameter are constrained using the Nelson & Cao specifications to ensure that conditional variances are nonnegative for all observations in and out of sample. Also, stationarity is assured by constraining roots to be outside unit circle. This involves a nonlinear constraint on parameters. These are the least restrictive constraints that satisfy the conditions of nonnegative conditional variances and stationarity.

bounds Nonnegativity of conditional variances is carried out by direct constraints on the conditional variances. This does not assure nonnegativity outside of the sample. Stationarity is imposed by placing bounds on parameters, that is, **arch** and **garch** coefficients are constrained to be greater than zero and sum to less than one. These constraints are more restrictive than the standard coefficients, and are the most commonly applied constraints.

unconstrained Conditional variances are directly constrained to be nonnegative as in the bounds method, but no constraints are applied to ensure stationarity.

Source

`fankeymt.src`

Purpose

Sets type of covariance matrix of parameters.

Library

fanpacmt

Format

setCovParType *type*;

Input

type String, type of covariance matrix.

ML Maximum likelihood.

XPROD Cross product of first derivatives.

QML Quasi-maximum likelihood.

Global Output

_fan_CovParType Scalar, type of covariance matrix of parameters.

ML Maximum likelihood.

XPROD Cross-product of first derivatives.

QML Quasi-maximum likelihood.

Remarks

let $H = \partial logl/\partial\theta\partial\theta'$ be the Hessian and $G = \partial logl/\partial\theta$ the matrix of first derivatives. Then ML $= H^{-1}$, XPROD $= (G'G)^{-1}$, and $WML = H^{-1}(G'G)H^{-1}$.

Source

fankeymt.src

Purpose

Declares independent variables for inclusion into conditional variance equation.

Library

`fanpacmt`

Format

setCVIndEqs *name list*;

Input

name	Name of time series for this set of independent. variables
list	List of names of independent variables.

Global Output

_fan_CVIndEquations	L×K character vector, names of independent variables for each equation.

Remarks

An equation is associated with each time series. For multivariate models, call **setCVIndEqs** for each time series, listing the independent variables by name in each call:

```
setCVIndEqs msft  logVol1  SandP
setCVIndEqs intc  logVol2  SandP
```

If time series names are omitted, only one call is permitted and all independent variables are assumed to be entered in all equations.

```
setCVIndEqs  logVol1 logVol2 SandP
```

Source

`fankeymt.src`

■ Purpose

Sets dataset name for analysis.

■ Library

`fanpacmt`

■ Format

setDataset *name* **[***newname***];**

■ Input

name Name of file containing data.

newname If *name* is not the name of a **GAUSS** data set, a **GAUSS** data set will be created with name *newname* from the data in *name*.

■ Global Input

_fan_VarNames Scalar or K×1 character vector, column numbers
– or –
variable names of the columns
of the data in the data file. If *name* is not a **GAUSS** data set file, **_fan_VarNames** is required to name the variables in the data set.

If **_fan_VarNames** is set to scalar number of columns, the variables in the data file will be given labels X1, X2..... If **_fan_VarNames** is scalar missing (default), it is assumed that the data file contains a single column of data.

■ Global Output

_fan_dataset String, name of **GAUSS** data set.

■ Remarks

If *name* is not a **GAUSS** data set file or a DRI database, **FANPACMT** assumes that *name* is a file containing the data.

If one of the columns in the **GAUSS** data set is labeled DATE, **FANPACMT** will assume that this variable is a date variable in the format *yyyymmddhhmmss*.

If the data file is not a **GAUSS** data set file or DRI database, and one of the variable names in **_fan_VarNames** is DATE, **FANPACMT** will assume that the associated column in the data on that file is a date variable. The format of the date in that file can be *mm/dd/yy* or *mm/dd/yyyy* or *yyyymmdd*, and it will be put by **FANPACMT** into the *yyyymmddhhmmss* format.

If the data in the data file are in the nonstandard order, i.e., from most recent date at the top to the oldest date at the bottom, **FANPACMT** reverses the order of the data in the **GAUSS** data set generated from the data. This will also occur if any of the dates are out of order. If the data are stored in a **GAUSS** data set, this check will not be made.

Example

```
library fanpacmt,pgraph;
session nissan 'Analysis of Nissan daily log-returns';
setVarNames date nsany;
setDataset nsany.asc;
setSeries nsany;
estimate run1 garch(1,3);
showResults;
```

Source

fankeymt.src

Purpose

Causes an independent variable measuring elapsed time between observations to be generated

Library

`fanpacmt`

Format

setIndElapsedPeriod;

Global Output

_fan_IndVars N×KL matrix, independent variables

Remarks

Adding this keyword command to the command file causes **FANPAC MT** to generate an independent variable measuring the elapsed time between observations. The name of this variable is **EP**. For example,

```
library fanpacmt,pgraph;
session wilshire 'wilshire example';
setDataset wilshire;
setSeries cwret; /* capitalization weighted returns */
setIndElapsedPeriod;
setIndEqs EP;

estimate run1 garch(2,2);

showResults;
```

Source

`fankeymt.src`

Purpose

Sets independent variable list for particular run.

Library

`fanpacmt`

Format

setIndEqs *name list*;

Input

name	Name of time series for this set of independent variables, may be omitted for univariate runs.
list	List of names of independent variables.

Global Output

_fan_IndEquations	L×K matrix, indicator matrix for coefficients to be estimated.

Remarks

For multivariate models, call **setIndEqs** for each time series, listing the independent variables by name in each call:

```
setIndEqs msft  lnVol1  SandP
setIndEqs intc  lnVol2  SandP
```

If **setIndEqs** is not called for a particular dependent variable, coefficients for all independent variables will be estimated for that dependent variable.

Source

`fankeymt.src`

■ Purpose

Causes an independent variable measuring log elapsed time between observations to be generated

■ Library

fanpacmt

■ Format

setIndLogElapsedPeriod;

■ Global Output

_fan_IndVars N×KL matrix, independent variables

■ Remarks

Adding this keyword command to the command file causes **FANPAC MT** to generate an independent variable measuring the elapsed time between observations. The name of this variable is **lnEP**. For example,

```
library fanpacmt,pgraph;
session wilshire 'wilshire example';
setDataset wilshire;
setSeries cwret; /* capitalization weighted returns */
setIndLogElapsedPeriod;
setIndEqs lnEP;

estimate run1 garch(2,2);

showResults;
```

■ Source

fankeymt.src

Purpose

Sets type of statistical inference.

Library

`fanpacmt`

Format

setInferenceType [*type*];

Input

type If omitted, standard errors computed from covariance matrix of parameters are computed. Otherwise, set to

NONE no confidence limits computed

SIMLIMITS confidence limits by Andrew's simulation method

WALD confidence limits computed from covariance matrix of parameters.

Remarks

Source

`fankeymt.src`

setInmean

Purpose

Declares inmean model

Library

`fanpacmt`

Format

setInmean;

Source

`fankeymt.src`

Purpose

Declares exogenous or independent variables.

Library

`fanpacmt`

Format

setIndVars *list*;

Input

list	List of names of independent variables for current session.

Global Output

_fan_IndvarNames	L×K character vector, names of independent variables for each equation.

Remarks

A variable in the list can be log-transformed by pre-pending an asterisk to the name in the list. For example

```
setIndVars *volume;
```

causes **FANPAC MT** to generate the independent variable `ln(volume)`. The label for the new variable is the old label pre-pended by "ln". This to add it to the list of independent variables for a particular run add

```
setIndEqs lnvolume;
```

to the command file prior to the call to **estimate**.

Source

`fankeymt.src`

■ Purpose

Sets number of lags INCLUDED in analysis for FIGARCH models.

■ Library

`fanpacmt`

■ Format

setLagTruncation *num*;

■ Input

num	Number of lags included.

■ Remarks

The conditional variance in the FIGARCH(p,q) model is the sum of an infinite series of prior conditional variances. In practice, the log-likelihood is computed from available data; and this means that the calculation of the conditional variance will be truncated. To minimize this error, the log-probabilities for initial observations can be excluded from the log-likelihood. The default is one-half of the observations. To change this specification, **setLagTruncation** can be set to some other value that determines the number of observations to be included.

■ Source

`fankeymt.src`

Purpose

Sets number of lags EXCLUDED in analysis for FIGARCH models.

Library

`fanpacmt`

Format

setLagInitialization *num*;

Input

num	Number of lags included.

Remarks

The conditional variance in the FIGARCH(p,q) model is the sum of an infinite series of prior conditional variances. In practice, the log-likelihood is computed from available data; and this means that the calculation of the conditional variance will be truncated. To minimize this error, the log-probabilities for initial observations can be excluded from the log-likelihood. The default is one-half of the observations. To change this specification **setLagInitialization** can be set to some other value that determines the number of observations to be excluded.

Source

`fankeymt.src`

■ Purpose

Sets order for Ljung-Box statistic.

■ Library

`fanpacmt`

■ Format

setLjungBoxOrder *order*;

■ Input

order	Number of autocorrelations included in the Ljung-Box test statistic. It must be less than the total number of observations.

■ Source

`fankeymt.src`

Purpose

Sets output file name and status.

Library

`fanpacmt`

Format

setOutputFile *filename* **[***action***];**

Input

filename	Output file is created with this name.
action	String. If absent, output file is turned on, otherwise, set to **ON** output file is turned on, **OFF** output file is turned off, **RESET** output file is reset.

Source

`fankeymt.src`

Purpose

sets range of time series to be analyzed

Library

`fanpacmt`

Format

setRange *start end*;

Input

start Scalar, starting row or date to be included in series. If row number, it must be greater than 1 and less than *end*.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *start* to START is equivalent to first observation.

end scalar, ending row or date to be included in series. If row number, it must be greater than *start* and less than or equal to the number of observations.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *end* to END is equivalent to last observation.

Global Output

_fan_Series N×L matrix, time series.

_fan_Date N/*times*1 vector, dates of observations in yyyymmmdd format. This requires that the session dataset contain a variable in that same format with variable name "date."

Source

`fankeymt.src`

Purpose

Declares time series to be analyzed.

Library

`fanpacmt`

Format

setSeries *list* **[***start end***];**

Input

list List of names of time series.

start Scalar, starting row or date to be included in series. If row number, it must be greater than 1 and less than *end*.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *start* to START is equivalent to first observation.

end Scalar, ending row or date to be included in series. If row number, it must be greater than *start* and less than or equal to the number of observations.

If date, it may be in one of the formats, *yyyymmdd*, *yyyymmddhhmmss*, *mm/dd/yy*, *mm/dd/yyyy*, where if yy the 20th century is assumed. The session dataset must also have included a variable with the variable name "date."

Setting *end* to END is equivalent to last observation.

Global Output

_fan_Series N×L matrix, time series.

_fan_SeriesNames L×1 character vector, names of time series.

_fan_Date N/*times*1 vector, dates of observations in yyyymmmdd format. This requires that the session dataset contain a variable in that same format with variable name "date."

Source

`fankeymt.src`

■ Purpose

Sets variable names for ASCII file containing data.

■ Library

`fanpacmt`

■ Format

setVarNames *list*;

■ Input

list	Variable names of the columns of an ASCII file containing data. – or – scalar number of columns of data in ASCII file

■ Global Output

_fan_VarNames	K×1 character vector, variable names of data in the ASCII data file.

■ Remarks

If list is a scalar number of columns, variables in data file will be given labels X1, X2,....

■ Source

`fankeymt.src`

Purpose

Displays estimates and their lables in a simple format

Library

`fanpacmt`

Format

showEstimates *list*;

Input

list	List of names of estimation runs. If no run names are provided, all runs are displayed.

Source

`fankeymt.src`

Keyword Reference

Purpose

Displays results of a run.

Library

`fanpacmt`

Format

showResults *list*;

Input

list	List of names of estimation runs. If no run names are provided, all runs are displayed.

Example

```
library fanpacmt,pgraph;

session test 'test session';

setDataset stocks;
setSeries intel;
setOutputfile test.out reset;

estimate run1 garch;
estimate run2 garch(2,1);
estimate run3 arima(1,2,1);

showResults;
```

Source

`fankeymt.src`

Purpose

Displays a List of current runs in a session.

Library

`fanpacmt`

Format

showRuns;

Source

`fankeymt.src`

Purpose

Simulates data with GARCH errors.

Library

`fanpacmt`

Format

simulate *starray*;

Input

starray K×1 string array, simulation parameters

Model model name (required).

NumObs number of observations (required).

DatasetName name of **GAUSS** data set into which simulated data will be put (required).

TimeSeriesName variable label of time series.

Omega GARCH process constant, required for GARCH models.

GarchCoefficients GARCH coefficients, required for GARCH models.

ArchCoefficients ARCH coefficients, required for GARCH models.

ARCoefficients AR coefficients, required for ARIMA models.

MACoefficients MA coefficients, required for ARIMA models.

RegCoefficients Regression coefficients, required for OLS models.

DFCoefficient degrees of freedom parameter for t-density. If set, t-density will be used; otherwise Normal density.

Constant constant (required).

Seed seed for random number generator (optional).

Example

```
library fanpacmt;

string ss = {

  "Model garch(1,2)",
  "NumObs 300",
  "DatasetName example",
  "TimeSeriesName Y",
```

```
    "Omega .2",
    "GarchParameter .5",
    "ArchParameter .4 -.1",
    "Constant .5",
    "Seed 7351143"
  };

  simulate ss;
```

Source

fansimmt.src

Keyword Reference

■ Purpose

Computes skew and kurtosis statistics and a heteroskedastic-consistent Ljung-Box statistic for standardized residuals as well as time series.

■ Library

`fanpacmt`

■ Format

testSR *list*;

■ Input

list	List of runs.

■ Remarks

The Ljung-Box statistic is the heteroskedastic-consistent statistic described in Gouriéroux, 1997.

■ Source

`fankeymt.src`

Chapter 4

FANPACMT Procedure Reference

Purpose

Computes conditional variances for the multivariate garch model

Library

`fanpacmt`

Format

r = **mcvar(**F,D**);**

Input

F	instance of a fanEstimation structure
D	1×1 or 2×1 instance of DS structure
	d0[1 .dataMatrix] $N \times M$ matrix, time series *d0[2* .dataMatrix] $N \times k$ matrix, independent variables (optional).

Output

r	$N \times 1$ vector, conditional variances

Source

`mgarchmt.src`

Purpose

Computes time series and conditional variance forecasts for multivariate time series.

Library

`fanpacmt`

Format

{ r,s } = **mforecast(**F**,**D**,**$period$**,**xp**);**

Input

F	instance of a fanEstimation structure
D	1×1 or 2×1 instance of DS structure *d0[1* .dataMatrix] $N \times M$ matrix, time series *d0[2* .dataMatrix] $N \times k$ matrix, independent variables (optional).
$period$	Scalar, number of periods to be forecast.
xp	$M \times K$ matrix, forecast independent variables. If there are independent variables but no forecast independent variables, set $xp = 0$, and the means of the independent variables will be used for forecast.

Output

r	$L \times M$ Matrix, L period forecast of times series.
s	$L \times M$ matrix or $L \times M \times M$ array, L period forecast of conditional variance.

Source

`mgarchmt.src`

Purpose

Estimates parameters of multivariate time series.

Library

`fanpacmt`

Format

out = mgarch(C,D**);**

Input

C instance of a fanControl structure

C.p scalar, order of the garch parameters

C.q scalar, order of the arch parameters

C.ar scalar, order of the autoregressive parameters

C.ma scalar, order of the moving average parameters

C.density scalar, density of error term, 0 - Normal, 1 - Student's t, 2 - generalized exponential

C.multModel scalar, 0 - diagonal vec, 1 - constant correlation diagonal vec, 2 - BEKK

C.fractional scalar, if nonzero, fractional integrated model

C.leverage scalar, if nonzero leverage terms are added

C.assymetry scalar, if nonzero assymetry terms are added

C.unitRoot scalar, if nonzero a unit root is forced on the determinantal polynomial

C.indEquations $M \times K$ matrix, a $1 \times K$ vector for each mean equation indicating which independent variables are included.

C.bxcx $1 \times K$ vector, mask indicating which independent variables are to be transformed via the boxcox function

C.seriesbxcx scalar, if nonzero the time series is transformed via the boxcox function

C.CVIndEquations $L \times K$ matrix, a $1 \times K$ vector for each conditional variance equation indicating independent variables to be included.

C.inMean scalar, $M \times L$ matrix, a $1 \times L$ vector for each mean equation indicating which conditional variance equation is included in which mean equation.

C.stConstType scalar, type of enforcement of stationarity requirements, 1 - roots of characteristic polynomial constrained outside unit circle, 2 - arch, garch parameters constrained to sum to less than one and greater than zero, 3 - none

C.cvConstType scalar, type of enforcement of nonnegative conditional variances, 0 - direct constraints, 1 - Nelson & Cao constraints

C.covType scalar, type of covariance matrix of parameters, 1 - ML, 2 - QML, 3 - none.

d0 1x1 or 2x1 DS structure, data.

d0[1].dataMatrix, Nx1 vector, time series

d0[2].dataMatrix, Nxk matrix, independent variables (optional).

Output

out instance of a fanEstimation structure

model scalar

- *0* OLS
- *1* ARIMA
- *2* GARCH
- *3* FIGARCH
- *4* IGARCH
- *5* EGARCH
- *10* SIMEQ
- *11* VARMA
- *12* DVGARCH
- *13* DVFIGARCH
- *14* CDVGARCH
- *15* CDVFIGARCH
- *16* CDVEGARCH
- *17* BKGARCH

control a copy of the input fanControl structure

aic scalar, scalar, Akiake criterion

bic scalar, Bayesian information criterion

lrs scalar, likelihood ratio statistic

numObs scalar, number of observations

df scalar, degress of freedom

par instance of PV structure containing parameter estimates

retcode scalar, return code

0 normal convergence
1 forced exit
2 maximum number of iterations exceeded
3 function calculation failed
4 gradient calculation failed
5 Hessian calculation failed
6 line search failed
7 error with constraints
8 function complex

moment KxK matrix, moment matrix of parameter estimates

climits Kx2 matrix, confidence limits

tsForecast MxL matrix, time series forecast

cvForecast MxLxL array, forecast of conditional covariance matrices

Source

`mgarchmt.src`

Purpose

Computes residuals for the multivariate garch model

Library

`fanpacmt`

Format

r = **mres(**F,D,s**);**

Input

F	instance of a fanEstimation structure
D	1×1 or 2×1 instance of DS structure
	D0[1 .dataMatrix] $N \times 1$ vector, time series
	D0[2 .dataMatrix] $N \times k$ matrix, independent variables (optional).
s	scalar, if nonzero standardized residuals are computed, otherwise they are unstandardized. Default = 0.

Output

r	$N \times 1$ vector, residuals

Source

`mgarchmt.src`

Purpose

Computes roots of the characteristic equation for multivariate models

Library

`fanpacmt`

Format

r = **mroots(**F**);**

Input

F	instance of a fanEstimation structure

Output

r	$L \times 1$ vector, roots.

Remarks

For the diagonal vec model **mroot** computes roots of

$$1 - [(\alpha_1 + \beta_1)]Z - [(\alpha_2 + \beta_2)]Z^2 + \cdots$$

$$1 - [\beta_1]Z - [\beta_2]Z^2 + \cdots + [\beta_p]Z^p$$

where α_i and β_i are $L(L+1)/2 \times 1$ vectors of the ARCH and GARCH parameters for the diagonal vec model and [] indicates expansion to symmetric matrices.

For the constant correlation diagonal vec model α_i and β_i $L \times 1$ vectors and [] indicates expansion to diagonal matrices.

For the BEKK model α_i and β_i are $L \times L$ matrices of parameters and [] indicates no change.

For all models, roots of the characteristic polynomial for the AR and MA parameters are also computed:

$$1 - \phi_1 Z - \phi_2 Z^2 + \cdots + \phi_p Z^p$$

$$1 - \theta_1 Z - \theta_2 Z^2 + \cdots + \theta_p Z^p$$

where the ϕ_i are the AR parameters, and where the θ_i are the MA parameters.

Source

`mgarchmt.src`

Purpose

simulates time series

Library

`fanpacmt`

Format

D = **mSimulation(**S**);**

Input

S instance of fanSimulation structure

s0.par instance of PV structure containing packed parameter matrices

beta0 $L \times 1$ vector, constants in mean equations
omega $L \times 1$ vector, constants in variance equations
garch $P \times L$ matrix, garch parameters
arch $Q \times L$ matrix, arch parameters
phi $R \times L \times L$ array, AR parameters
theta $S \times L \times L$ array, MA parameters
tau $L \times 1$ vector, asymmetry parameters
delta $L \times 1$ vector, inmean coefficients, variance
delta_s $L \times 1$ vector, inmean coefficients, standard dev

s0.numObs scalar, number of observations

s0.seed scalar, seed for random number generator

Output

D 1×1 or 2×1 instance of DS structure

D0[1 .dataMatrix] $N \times L$ vector, time series

Remarks

Parameters are specified by packing the appropriate matrices into S.par using the **pvPackm** functions. For example,

```
struct PV p0;
struct fanSimulation s0;

p0 = pvPack(p0,.5~.6,"garch");
p0 = pvPack(p0,.3~.4,"arch");
p0 = pvPack(p0,.5|.4,"beta0");
p0 = pvPack(p0,1|1,"omega");

s0.par = p0;
s0.numObs = 100;

struct DS d0;
d0 = mSimulation(s0);
```

Source

mgarchmt.src

Purpose

Statistical inference using Andrews simulation method

Library

`fanpacmt`

Format

cl,*vc* = **simlimits(***&fnct*,*S*,*D*,*F***);**

Input

&fnct	scalar, pointer to procedure computing minus log-likelihood
F	instance of an sqpSolveMTout structure containing results of estimation
D	1×1 or 2×1 instance of DS structure containing data used for estimation *D0[1* .dataMatrix] $N \times 1$ vector, time series *D0[2* .dataMatrix] $N \times k$ matrix, independent variables (optional).
F	instance of fanControl structure containing control variables used in estimation

Output

cl	$K \times 2$ vector, lower and upper confidence limits
vc	$K \times K$ matrix, covariance matrix

Source

`simlimitsmt.src`

Procedure Reference

■ Purpose

Computes conditional variances for the univariate garch model

■ Library

`fanpacmt`

■ Format

r = **ucvar(**F,D**);**

■ Input

F	instance of a fanEstimation structure
D	1×1 or 2×1 instance of DS structure
	D0[1 .dataMatrix] $N \times 1$ vector, time series
	D0[2 .dataMatrix] $N \times k$ matrix, independent variables (optional).

■ Output

r	$N \times 1$ vector, conditional variances

■ Source

`ugarchmt.src`

Purpose

Computes time series and conditional variance forecasts for univariate time series.

Library

`fanpacmt`

Format

{ r,s } = **uforecast**(F,D,$period$,xp);

Input

F	instance of a fanEstimation structure
P	instance of PV structure
D	1×1 or 2×1 instance of DS structure
	D0[1 .dataMatrix] $N \times 1$ vector, time series
	D0[2 .dataMatrix] $N \times k$ matrix, independent variables (optional).
period	Scalar, number of periods to be forecast.
xp	$M \times K$ matrix, forecast independent variables. If there are independent variables but no forecast independent variables, set $xp = 0$, and the means of the independent variables will be used for forecast.

Output

r	$M \times 1$ vector, M period forecast of times series.
s	$M \times 1$ vector, M period forecast of conditional variance.

Source

`ugarchmt.src`

Procedure Reference

Purpose

Estimates parameters of univariate time series.

Library

`fanpacmt`

Format

out = ugarch(C**,**D**);**

Input

C instance of a fanControl structure

C.p scalar, order of the garch parameters

C.q scalar, order of the arch parameters

C.ar scalar, order of the autoregressive parameters

C.ma scalar, order of the moving average parameters

C.density scalar, density of error term, 0 - Normal, 1 - Student's t, 2 - generalized exponential, 3 - skew generalized t

C.multModel scalar, 0 - diagonal vec, 1 - constant correlation diagonal vec, 2 - BEKK

C.fractional scalar, if nonzero, fractional integrated model

C.leverage scalar, if nonzero leverage terms are added

C.assymetry scalar, if nonzero assymetry terms are added

C.unitRoot scalar, if nonzero a unit root is forced on the determinantal polynomial

C.indEquations $M \times K$ matrix, a $1 \times K$ vector for each mean equation indicating which independent variables are included.

C.bxcx $1 \times K$ vector, mask indicating which independent variables are to be transformed via the boxcox function

C.seriesbxcx scalar, if nonzero the time series is transformed via the boxcox function

C.CVIndEquations $L \times K$ matrix, a $1 \times K$ vector for each conditional variance equation indicating independent variables to be included.

C.inMean scalar, $M \times L$ matrix, a $1 \times L$ vector for each mean equation indicating which conditional variance equation is included in which mean equation.

C.stConstType scalar, type of enforcement of stationarity requirements, 1 - roots of characteristic polynomial constrained outside unit circle, 2 - arch, garch parameters constrained to sum to less than one and greater than zero, 3 - none

C.cvConstType scalar, type of enforcement of nonnegative conditional variances, 0 - direct constraints, 1 - Nelson & Cao constraints

C.covType scalar, type of covariance matrix of parameters, 1 - ML, 2 - QML, 3 - none.

D 1x1 or 2x1 DS structure, data.

D[1].dataMatrix, Nx1 vector, time series

D[2].dataMatrix, Nxk matrix, independent variables (optional).

Output

out instance of a fanEstimation structure

model scalar

- *0* OLS
- *1* ARIMA
- *2* GARCH
- *3* FIGARCH
- *4* IGARCH
- *5* EGARCH
- *10* SIMEQ
- *11* VARMA
- *12* DVGARCH
- *13* DVFIGARCH
- *14* CDVGARCH
- *15* CDVFIGARCH
- *16* CDVEGARCH
- *17* BKGARCH

control a copy of the input fanControl structure

aic scalar, scalar, Akiake criterion

bic scalar, Bayesian information criterion

lrs scalar, likelihood ratio statistic

numObs scalar, number of observations

df scalar, degress of freedom

par instance of PV structure containing parameter estimates

retcode scalar, return code

0 normal convergence
1 forced exit
2 maximum number of iterations exceeded
3 function calculation failed
4 gradient calculation failed
5 Hessian calculation failed
6 line search failed
7 error with constraints
8 function complex

moment KxK matrix, moment matrix of parameter estimates

climits Kx2 matrix, confidence limits

tsForecast Mx1 vector, time series forecast

cvForecast Mx1x1 array, forecast of conditional variances

Source

`ugarchmt.src`

Purpose

Computes residuals for the univariate garch model

Library

`fanpacmt`

Format

r = **uRes(**F,D,s**);**

Input

F	instance of a fanEstimation structure
D	1×1 or 2×1 instance of DS structure
	D0[1 .dataMatrix] $N \times 1$ vector, time series
	D0[2 .dataMatrix] $N \times k$ matrix, independent variables (optional).
s	scalar, if nonzero standardized residuals are computed, otherwise they are unstandardized. Default = 0.

Output

r	$N \times 1$ vector, residuals

Source

`ugarchmt.src`

Purpose

Computes roots of the characteristic polynomial for univariate models

Library

`fanpacmt`

Format

r = **uRoots(**F**);**

Input

F	instance of a fanEstimation structure

Output

r	$L \times 1$ vector, roots.

Remarks

Computes roots of

$$1 - (\alpha_1 + \beta_1)Z - (\alpha_2 + \beta_2)Z^2 + \cdots$$

$$1 - \beta_1 Z - \beta_2 Z^2 + \cdots + \beta_p Z^p$$

and

$$1 - \phi_1 Z - \phi_2 Z^2 + \cdots + \phi_p Z^p$$

$$1 - \theta_1 Z - \theta_2 Z^2 + \cdots + \theta_p Z^p$$

where the β_i are the GARCH parameters, where the α_i are the ARCH parameters, where the ϕ_i are the AR parameters, and where the θ_i are the MA parameters.

Source

`ugarchmt.src`

Purpose

simulates univariate time series

Library

`fanpacmt`

Format

D = **uSimulation(**S**);**

Input

S instance of fanSimulation structure

s0.par instance of PV structure containing packed parameter matrices

beta0 scalar, constant in mean equations
omega scalar, constant in variance equations
garch $P \times 1$ vector, garch parameters
arch $Q \times 1$ vector, arch parameters
phi $R \times 1$ vector, AR parameters
theta $S \times 1$ vector, MA parameters
tau scalar, asymmetry parameter
delta scalar, inmean coefficient, variance
delta_s scalar, inmean coefficient, standard dev

s0.numObs scalar, number of observations

s0.seed scalar, seed for random number generator

Output

D 1×1 or 2×1 instance of DS structure

D0[1 .dataMatrix] $N \times L$ vector, time series

Remarks

Parameters are specified by packing the appropriate matrices into S.par using the **pvPackm** functions. For example,

Procedure Reference

```
struct PV p0;
struct fanSimulation s0;

p0 = pvPack(p0,.5,"garch");
p0 = pvPack(p0,.3,"arch");
p0 = pvPack(p0,.5,"beta0");
p0 = pvPack(p0,1,"omega");

s0.par = p0;
s0.numObs = 100;

struct DS d0;
d0 = uSimulation(s0);
```

Source

ugarchmt.src

Index