Package 'fGarch'

November 5, 2022

Title Rmetrics - Autoregressive Conditional Heteroskedastic Modelling

Version 4022.89

Description Analyze and model heteroskedastic behavior in financial time series.

Imports fBasics, timeDate, timeSeries, fastICA, Matrix (>= 1.5-0), cvar (>= 0.5), graphics, methods, stats, utils

Suggests RUnit, tcltk

LazyData yes

License GPL (>= 2)

URL https://www.rmetrics.org

BugReports https://r-forge.r-project.org/projects/rmetrics

NeedsCompilation yes

Author Diethelm Wuertz [aut] (original code), Yohan Chalabi [aut], Tobias Setz [aut], Martin Maechler [aut] (<https://orcid.org/0000-0002-8685-9910>), Chris Boudt [ctb], Pierre Chausse [ctb], Michal Miklovac [ctb], Georgi N. Boshnakov [cre, ctb]

Maintainer Georgi N. Boshnakov <georgi.boshnakov@manchester.ac.uk>

Repository CRAN

Date/Publication 2022-11-05 22:10:02 UTC

R topics documented:

`fGarch-package` *Modelling heterskedasticity in financial time series*

Description

The Rmetrics **fGarch** package is a collection of functions to analyze and model heteroskedastic behavior in financial time series.

1 Introduction

GARCH, Generalized Autoregressive Conditional Heteroskedastic, models have become important in the analysis of time series data, particularly in financial applications when the goal is to analyze and forecast volatility.

For this purpose, the family of GARCH functions offers functions for simulating, estimating and forecasting various univariate GARCH-type time series models in the conditional variance and an ARMA specification in the conditional mean. The function `garchFit` is a numerical implementation of the maximum log-likelihood approach under different assumptions, Normal, Student-t, GED

errors or their skewed versions. The parameter estimates are checked by several diagnostic analysis tools including graphical features and hypothesis tests. Functions to compute n-step ahead forecasts of both the conditional mean and variance are also available.

The number of GARCH models is immense, but the most influential models were the first. Beside the standard ARCH model introduced by Engle [1982] and the GARCH model introduced by Bollerslev [1986], the function `garchFit` also includes the more general class of asymmetric power ARCH models, named APARCH, introduced by Ding, Granger and Engle [1993]. The APARCH models include as special cases the TS-GARCH model of Taylor [1986] and Schwert [1989], the GJR-GARCH model of Glosten, Jaganathan, and Runkle [1993], the T-ARCH model of Zakoian [1993], the N-ARCH model of Higgins and Bera [1992], and the Log-ARCH model of Geweke [1986] and Pentula [1986].

There exist a collection of review articles by Bollerslev, Chou and Kroner [1992], Bera and Higgins [1993], Bollerslev, Engle and Nelson [1994], Engle [2001], Engle and Patton [2001], and Li, Ling and McAleer [2002] which give a good overview of the scope of the research.

2 Time series simulation

Functions to simulate artificial GARCH and APARCH time series processes.

`garchSpec`	specifies an univariate GARCH time series model
`garchSim`	simulates a GARCH/APARCH process

3 Parameter estimation

Functions to fit the parameters of GARCH and APARCH time series processes.

`garchFit`	fits the parameters of a GARCH process

Extractor Functions::

`residuals`	extracts residuals from a fitted `"fGARCH"` object
`fitted`	extracts fitted values from a fitted `"fGARCH"` object
`volatility`	extracts conditional volatility from a fitted `"fGARCH"` object
`coef`	extracts coefficients from a fitted `"fGARCH"` object
`formula`	extracts formula expression from a fitted `"fGARCH"` object

4 Forecasting

Functions to forcecast mean and variance of GARCH and APARCH processes.

`predict`	forecasts from an object of class `"fGARCH"`

5 Standardized distributions

This section contains functions to model standardized distributions.

Skew normal distribution::

`[dpqr]norm`	Normal distribution (base R)

[dpqr]snorm	Skew normal distribution
snormFit	fits parameters of Skew normal distribution

Skew generalized error distribution::

[dpqr]ged	Generalized error distribution
[dpqr]sged	Skew Generalized error distribution
gedFit	fits parameters of Generalized error distribution
sgedFit	fits parameters of Skew generalized error distribution

Skew standardized Student-t distribution::

[dpqr]std	Standardized Student-t distribution
[dpqr]sstd	Skew standardized Student-t distribution
stdFit	fits parameters of Standardized Student-t distribution
sstdFit	fits parameters of Skew standardized Student-t distribution

Absolute moments::

absMoments	computes absolute moments of these distribution

About Rmetrics

The fGarch Rmetrics package is written for educational support in teaching "Computational Finance and Financial Engineering" and licensed under the GPL.

Author(s)

Diethelm Wuertz [aut] (original code), Yohan Chalabi [aut], Tobias Setz [aut], Martin Maechler [ctb] (<https://orcid.org/0000-0002-8685-9910>), Chris Boudt [ctb] Pierre Chausse [ctb], Michal Miklovac [ctb], Georgi N. Boshnakov [cre, ctb]

Maintainer: Georgi N. Boshnakov <georgi.boshnakov@manchester.ac.uk>

---

absMoments *Absolute moments of GARCH distributions*

Description

Computes absolute moments of the standard normal, standardized GED, and standardized skew Student-t distributions.

Usage

```
absMoments(n, density = c("dnorm", "dged", "dstd"), ...)
```

Arguments

n	the order of the absolute moment, can be a vector to compute several absolute moments at once.
density	a character string naming a symmetric density function.
...	parameters passed to the density function.

Details

absMoments returns a numeric vector of length n with the values of the absolute moments, as specified by n, of the selected probability density function (pdf).

If density names one of the densities in the signature of absMoments, the moments are calculated from known formulas.

Otherwise, numerical integration is used and an attribute is attached to the results to report an estimate of the error. Note that the density is assumed symmetric wihtout a check.

Value

a numeric vector

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

ged, std

Examples

```
## absMoment -

absMoments(1, "dstd", nu = 6)
absMoments(1, "dstd", nu = 600)
absMoments(1, "dstd", nu = 60000)
absMoments(1, "dstd", nu = 600000)

absMoments(1, "dnorm")

## excess kurtosis of t_nu is  6/(nu - 4)
nu <- 6
absMoments(2*2, "dstd", nu = nu) / absMoments(2*1, "dstd", nu = nu)^2 - 3
6/(nu-4)

## 4th moment for t_4 is infinite
```

```
absMoments(4, "dstd", nu = 4)

absMoments(1, "dged", nu = 4)
```

`coef-methods` *GARCH coefficients methods*

Description

Coefficients methods `coef()` for GARCH Models.

Methods

Methods for `coef` defined in package **fGarch**:

object = "fGARCH" Extractor function for coefficients from a fitted GARCH model.

object = "fGARCHSPEC" Extractor function for coefficients from a GARCH specification structure.

Note

`coef` is a generic function which extracts coefficients from objects returned by modeling functions.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

Examples

```
## garchSpec -
   # Use default parameters beside alpha:
   spec = garchSpec(model = list(alpha = c(0.05, 0.05)))
   spec
   coef(spec)

## garchSim -
   # Simulate an univariate "timeSeries" series from specification 'spec':
   x = garchSim(spec, n = 2000)
   x = x[,1]

## garchFit --
   fit = garchFit( ~ garch(1, 1), data = x, trace = FALSE)

## coef -
   coef(fit)
```

fGARCH-class *Class "fGARCH"*

Description

The class 'fGARCH' represents a model of an heteroskedastic time series process.

Objects from the Class

Objects can be created by calls of the function `garchFit`. This object is a parameter estimate of an empirical GARCH process.

Slots

`call`: Object of class `"call"`: the call of the `garch` function.

`formula`: Object of class `"formula"`: a formula object specifying the mean and variance equations.

`method`: Object of class `"character"`: a string denoting the optimization method, by default `"Max Log-Likelihood Estimation"`.

`data`: Object of class `"list"`: a list with one entry named `x`, containing the data of the time series to be estimated, the same as given by the input argument `series`.

`fit`: Object of class `"list"`: a list with the results from the parameter estimation. The entries of the list depend on the selected algorithm, see below.

`residuals`: Object of class `"numeric"`: a numeric vector with the (raw, unstandardized) residual values.

`fitted`: Object of class `"numeric"`: a numeric vector with the fitted values.

`h.t`: Object of class `"numeric"`:
a numeric vector with the conditional variances ($h_t = \sigma_t^\delta$).

`sigma.t`: Object of class `"numeric"`: a numeric vector with the conditional standard deviations.

`title`: Object of class `"character"`: a title string.

`description`: Object of class `"character"`: a string with a brief description.

Methods

plot `signature(x = "fGARCH", y = "missing")`: plots an object of class `"fGARCH"`.

show `signature(object = "fGARCH")`: prints an object of class `"fGARCH"`.

summary `signature(object = "fGARCH")`: summarizes an object of class `"fGARCH"`.

predict `signature(object = "fGARCH")`: forecasts mean and volatility from an object of class `"fGARCH"`.

fitted `signature(object = "fGARCH")`: extracts fitted values from an object of class `"fGARCH"`.

residuals `signature(object = "fGARCH")`: extracts fresiduals from an object of class `"fGARCH"`.

volatility `signature(object = "fGARCH")`: extracts conditional volatility from an object of class `"fGARCH"`.

coef `signature(object = "fGARCH")`: extracts fitted coefficients from an object of class `"fGARCH"`.

formula `signature(x = "fGARCH")`: extracts formula expression from an object of class `"fGARCH"`.

Author(s)

Diethelm Wuertz and Rmetrics Core Team

See Also

garchFit, garchSpec, garchFitControl

Examples

```
## simulate a time series, fit a GARCH(1,1) model, and show it:
x <- garchSim( garchSpec(), n = 500)
fit <- garchFit(~ garch(1, 1), data = x, trace = FALSE)
fit # == print(fit) and also == show(fit)
```

---

fGarchData *Time series datasets*

Description

Datasets used in the examples, including DEM/GBP foreign exchange rates and data on SP500 index.

Format

dem2gbp is a data frame with one column "DEM2GBP" and 1974 rows (observations).

sp500dge is a data frame with one column "SP500DGE" and 17055 rows (observations).

Details

The data represent retuns. No further details have been recorded.

Further datasets are available in the packages that **fGarch** imports, see fBasicsData and TimeSeriesData.

Examples

```
data(dem2gbp)
head(dem2gbp)
tail(dem2gbp)
str(dem2gbp)
plot(dem2gbp[[1]])

data(sp500dge)
head(sp500dge)
tail(sp500dge)
str(sp500dge)
plot(sp500dge[[1]])

## list time series datasets from available in some packages
data(package = "fBasics")
data(package = "timeSeries")
```

fGARCHSPEC-class *Class "fGARCHSPEC"*

Description

Specification structure for an univariate GARCH time series model.

Objects from the Class

Objects can be created by calls of the function garchSpec. This object is specifies the parameters of an empirical GARCH process.

Slots

call: Object of class "call": the call of the garch function.

formula: Object of class "formula": a list with two formula entries for the mean and variance equation.

model: Object of class "list": a list with the model parameters.

presample: Object of class "matrix": a numeric matrix with presample values.

distribution: Object of class "character": a character string with the name of the conditional distribution.

rseed: Object of class "numeric": an integer with the random number generator seed.

Methods

show signature(object = "fGARCHSPEC"): prints an object of class 'fGARCHSPEC'.

Note

With Rmetrics Version 2.6.1 the class has been renamed from "garchSpec" to "fGARCHSPEC".

Author(s)

Diethelm Wuertz for the Rmetrics R-port

Examples

```
## garchSpec -
   spec = garchSpec()
   spec # print() or show() it
```

`fitted-methods` *Extract GARCH model fitted values*

Description

Extracts fitted values from a fitted GARCH object.

Details

`fitted()` is a generic function which extracts fitted values from objects returned by modeling functions. `methods(fitted)` will show the methods available in the current R session.

The method for `"fGARCH"` objects extracts the `@fitted` value slot from an object of class `"fGARCH"` as returned by the function `garchFit`.

The class of the returned value depends on the input to the function `garchFit` who created the object. The returned value is always of the same class as the input object to the argument `data` in the function `garchFit`, i.e. if you fit a `"timeSeries"` object, you will get back from the function `fitted` also a `"timeSeries"` object, if you fit an object of class `"zoo"`, you will get back again a `"zoo"` object. The same holds for a `"numeric"` vector, for a `"data.frame"`, and for objects of class `"ts"`, `"mts"`.

In contrast, the slot itself returns independent of the class of the data input always a numeric vector, i.e. the function call `rslot(object, "fitted")` will return a numeric vector.

Methods

Methods for `fitted` defined in package **fGarch**:

object = "fGARCH" Extractor function for fitted values.

Note

(GNB) Contrary to the description of the returned value of the `"fGARCH"` method, it is always `"numeric"`.

TODO: either implement the documented behaviour or fix the documentation.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

`predict`, `residuals`, `garchFit`, class `fGARCH`,

`plot`

Examples

```
stopifnot(require("timeSeries"))
## Swiss Pension fund Index -
   x = as.timeSeries(data(LPP2005REC))

## garchFit -
   # Fit LPP40 Bechmark:
   fit = garchFit(LPP40 ~ garch(1, 1), data = 100*x, trace = FALSE)
   fit

## fitted -
   # Fitted values are now a "timeSeries" oject:
   fitted = fitted(fit)
   head(fitted)
   class(fitted)

## slot -
   # The slot contains a numeric Vector:
   fitted = slot(fit, "fitted")
   head(fitted)
   class(fitted)
```

`formula-methods` *Extract GARCH model formula*

Description

Extracts formula from a formula GARCH object.

Details

`formula` is a generic function which extracts the formula expression from objects returned by modeling functions.

The `"fGARCH"` method extracts the `@formula` expression slot from an object of class `"fGARCH"` as returned by the function `garchFit`.

The returned formula has always a left hand side. If the argument `data` was an univariate time series and no name was specified to the series, then the left hand side is assigned the name of the data.set. In the multivariate case the rectangular `data` object must always have column names, otherwise the fitting will be stopped with an error message

The class of the returned value depends on the input to the function `garchFit` who created the object. The returned value is always of the same class as the input object to the argument `data` in the function `garchFit`, i.e. if you fit a `"timeSeries"` object, you will get back from the function `fitted` also a `"timeSeries"` object, if you fit an object of class `"zoo"`, you will get back again a `"zoo"` object. The same holds for a `"numeric"` vector, for a `"data.frame"`, and for objects of class `"ts"`, `"mts"`.

In contrast, the slot itself returns independent of the class of the data input always a numeric vector, i.e. the function call `rslot(object, "fitted")` will return a numeric vector.

Methods

Methods for `formula` defined in package **fGarch**:

object = "fGARCH" Extractor function for formula expression.

Note

(GNB) Contrary to the description of the returned value of the `"fGARCH"` method, it is always `"numeric"`.

TODO: either implement the documented behaviour or fix the documentation.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

`garchFit`, class `fGARCH`

Examples

```
## garchFit -
   fit = garchFit(~garch(1, 1), data = garchSim(), trace = FALSE)

## formula -
   formula(fit)

## A Bivariate series and mis-specified formula:
   x = garchSim(n = 500)
   y = garchSim(n = 500)
   z = cbind(x, y)
   colnames(z)
   class(z)
   ## Not run:
   garchFit(z ~garch(1, 1), data = z, trace = FALSE)

## End(Not run)
   # Returns:
   # Error in .garchArgsParser(formula = formula, data = data, trace = FALSE) :
   #   Formula and data units do not match.

## Doubled column names in data set - formula can't fit:
   colnames(z) <- c("x", "x")
   z[1:6,]
   ## Not run:
   garchFit(x ~garch(1, 1), data = z, trace = FALSE)

## End(Not run)
   # Again the error will be noticed:
   # Error in garchFit(x ~ garch(1, 1), data = z) :
   #   Column names of data are not unique.
```

```
## Missing column names in data set - formula can't fit:
   z.mat <- as.matrix(z)
   colnames(z.mat) <- NULL
   z.mat[1:6,]
   ## Not run:
   garchFit(x ~ garch(1, 1), data = z.mat, trace = FALSE)

## End(Not run)
   # Again the error will be noticed:
   # Error in .garchArgsParser(formula = formula, data = data, trace = FALSE) :
   #   Formula and data units do not match
```

---

fUGARCHSPEC-class *Class 'fUGARCHSPEC'*

Description

Class 'fUGARCHSPEC'.

Objects from the Class

Objects can be created by calls of the form `new("fUGARCHSPEC", ...)`.

Slots

`model`: Object of class `"list"` ~~

`distribution`: Object of class `"list"` ~~

`optimization`: Object of class `"list"` ~~

`documentation`: Object of class `"list"` ~~

Methods

No methods defined with class `"fUGARCHSPEC"` in the signature.

Note

(GNB) This class seems to be meant for internal use by the package.

See Also

class `"fGARCH"`

Examples

```
showClass("fUGARCHSPEC")
```

garchFit *Univariate or multivariate GARCH time series fitting*

Description

Estimates the parameters of a univariate ARMA-GARCH/APARCH process, or — experimentally — of a multivariate GO-GARCH process model. The latter uses an algorithm based on `fastICA()`, inspired from Bernhard Pfaff's package **gogarch**.

Usage

```
garchFit(formula = ~ garch(1, 1), data,
init.rec = c("mci", "uev"),
delta = 2, skew = 1, shape = 4,
cond.dist = c("norm", "snorm", "ged", "sged",
                      "std", "sstd", "snig", "QMLE"),
include.mean = TRUE, include.delta = NULL, include.skew = NULL,
        include.shape = NULL,
        leverage = NULL, trace = TRUE,

algorithm = c("nlminb", "lbfgsb", "nlminb+nm", "lbfgsb+nm"),
hessian = c("ropt", "rcd"),
        control = list(),
        title = NULL, description = NULL, ...)

garchKappa(cond.dist = c("norm", "ged", "std", "snorm", "sged", "sstd", "snig"),
           gamma = 0, delta = 2, skew = NA, shape = NA)

.gogarchFit(formula = ~garch(1, 1), data, init.rec = c("mci", "uev"),
            delta = 2, skew = 1, shape = 4,
            cond.dist = c("norm", "snorm", "ged", "sged",
                          "std", "sstd", "snig", "QMLE"),
            include.mean = TRUE, include.delta = NULL, include.skew = NULL,
            include.shape = NULL,
            leverage = NULL, trace = TRUE,
            algorithm = c("nlminb", "lbfgsb", "nlminb+nm", "lbfgsb+nm"),
            hessian = c("ropt", "rcd"),
            control = list(),
            title = NULL, description = NULL, ...)
```

Arguments

`algorithm`	a string parameter that determines the algorithm used for maximum likelihood estimation.
`cond.dist`	a character string naming the desired conditional distribution. Valid values are `"dnorm"`, `"dged"`, `"dstd"`, `"dsnorm"`, `"dsged"`, `"dsstd"` and `"QMLE"`. The default value is the normal distribution. See Details for more information.

control	control parameters, the same as used for the functions from nlminb, and 'bfgs' and 'Nelder-Mead' from optim.
data	an optional timeSeries or data frame object containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which armaFit is called. If data is an univariate series, then the series is converted into a numeric vector and the name of the response in the formula will be neglected.
delta	a numeric value, the exponent delta of the variance recursion. By default, this value will be fixed, otherwise the exponent will be estimated together with the other model parameters if include.delta=FALSE.
description	optional character string with a brief description.
formula	formula object describing the mean and variance equation of the ARMA-GARCH/APARCH model. A pure GARCH(1,1) model is selected e.g., for formula = ~garch(1,1). To specify an ARMA(2,1)-APARCH(1,1) process, use ~ arma(2,1) + aparch(1,1).
gamma	APARCH leverage parameter entering into the formula for calculating the expectation value.
hessian	a string denoting how the Hessian matrix should be evaluated, either hessian ="rcd", or "ropt". The default, "rcd" is a central difference approximation implemented in R and "ropt" uses the internal R function optimhess.
include.delta	a logical determining if the parameter for the recursion equation delta will be estimated or not. If false, the shape parameter will be kept fixed during the process of parameter optimization.
include.mean	this flag determines if the parameter for the mean will be estimated or not. If include.mean=TRUE this will be the case, otherwise the parameter will be kept fixed durcing the process of parameter optimization.
include.shape	a logical flag which determines if the parameter for the shape of the conditional distribution will be estimated or not. If include.shape=FALSE then the shape parameter will be kept fixed during the process of parameter optimization.
include.skew	a logical flag which determines if the parameter for the skewness of the conditional distribution will be estimated or not. If include.skew=FALSE then the skewness parameter will be kept fixed during the process of parameter optimization.
init.rec	a character string indicating the method how to initialize the mean and varaince recursion relation.
leverage	a logical flag for APARCH models. Should the model be leveraged? By default leverage=TRUE.
shape	a numeric value, the shape parameter of the conditional distribution.
skew	a numeric value, the skewness parameter of the conditional distribution.
title	a character string which allows for a project title.
trace	a logical flag. Should the optimization process of fitting the model parameters be printed? By default trace=TRUE.
...	additional arguments to be passed.

Details

"QMLE" stands for Quasi-Maximum Likelihood Estimation, which assumes normal distribution and uses robust standard errors for inference. Bollerslev and Wooldridge (1992) proved that if the mean and the volatility equations are correctly specified, the QML estimates are consistent and asymptotically normally distributed. However, the estimates are not efficient and "the efficiency loss can be marked under asymmetric ... distributions" (Bollerslev and Wooldridge (1992), p. 166). The robust variance-covariance matrix of the estimates equals the (Eicker-White) sandwich estimator, i.e.

$$V = H^{-1}G'GH^{-1},$$

where V denotes the variance-covariance matrix, H stands for the Hessian and G represents the matrix of contributions to the gradient, the elements of which are defined as

$$G_{t,i} = \frac{\partial l_t}{\partial \zeta_i},$$

where t_t is the log likelihood of the t-th observation and ζ_i is the i-th estimated parameter. See sections 10.3 and 10.4 in Davidson and MacKinnon (2004) for a more detailed description of the robust variance-covariance matrix.

Value

for garchFit, an S4 object of class "fGARCH". Slot @fit contains the results from the optimization.

for .gogarchFit(): Similar definition for GO-GARCH modeling. Here, data must be *multivariate*. Still "preliminary", mostly undocumented, and untested(!). At least mentioned here...

Author(s)

Diethelm Wuertz for the Rmetrics R-port,
R Core Team for the 'optim' R-port,
Douglas Bates and Deepayan Sarkar for the 'nlminb' R-port,
Bell-Labs for the underlying PORT Library,
Ladislav Luksan for the underlying Fortran SQP Routine,
Zhu, Byrd, Lu-Chen and Nocedal for the underlying L-BFGS-B Routine.

Martin Maechler for cleaning up; *mentioning* .gogarchFit().

References

ATT (1984); *PORT Library Documentation*, http://netlib.bell-labs.com/netlib/port/.

Bera A.K., Higgins M.L. (1993); *ARCH Models: Properties, Estimation and Testing*, J. Economic Surveys 7, 305–362.

Bollerslev T. (1986); *Generalized Autoregressive Conditional Heteroscedasticity*, Journal of Econometrics 31, 307–327.

Bollerslev T., Wooldridge J.M. (1992); *Quasi-Maximum Likelihood Estimation and Inference in Dynamic Models with Time-Varying Covariance*, Econometric Reviews 11, 143–172.

Byrd R.H., Lu P., Nocedal J., Zhu C. (1995); *A Limited Memory Algorithm for Bound Constrained Optimization*, SIAM Journal of Scientific Computing 16, 1190–1208.

Davidson R., MacKinnon J.G. (2004); *Econometric Theory and Methods*, Oxford University Press, New York.

Engle R.F. (1982); *Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation*, Econometrica 50, 987–1008.

Nash J.C. (1990); *Compact Numerical Methods for Computers*, Linear Algebra and Function Minimisation, Adam Hilger.

Nelder J.A., Mead R. (1965); *A Simplex Algorithm for Function Minimization*, Computer Journal 7, 308–313.

Nocedal J., Wright S.J. (1999); *Numerical Optimization*, Springer, New York.

See Also

garchSpec, garchFitControl, class "fGARCH"

Examples

```
## UNIVARIATE TIME SERIES INPUT:
   # In the univariate case the lhs formula has not to be specified ...

   # A numeric Vector from default GARCH(1,1) - fix the seed:
   N = 200
   x.vec = as.vector(garchSim(garchSpec(rseed = 1985), n = N)[,1])
   garchFit(~ garch(1,1), data = x.vec, trace = FALSE)

   # An univariate timeSeries object with dummy dates:
   stopifnot(require("timeSeries"))
   x.timeSeries = dummyDailySeries(matrix(x.vec), units = "GARCH11")
   garchFit(~ garch(1,1), data = x.timeSeries, trace = FALSE)

## Not run:
   # An univariate zoo object:
   require("zoo")
   x.zoo = zoo(as.vector(x.vec), order.by = as.Date(rownames(x.timeSeries)))
   garchFit(~ garch(1,1), data = x.zoo, trace = FALSE)

## End(Not run)

   # An univariate "ts" object:
   x.ts = as.ts(x.vec)
   garchFit(~ garch(1,1), data = x.ts, trace = FALSE)

## MULTIVARIATE TIME SERIES INPUT:
   # For multivariate data inputs the lhs formula must be specified ...

   # A numeric matrix binded with dummy random normal variates:
   X.mat = cbind(GARCH11 = x.vec, R = rnorm(N))
   garchFit(GARCH11 ~ garch(1,1), data = X.mat)
```

```
   # A multivariate timeSeries object with dummy dates:
   X.timeSeries = dummyDailySeries(X.mat, units = c("GARCH11", "R"))
   garchFit(GARCH11 ~ garch(1,1), data = X.timeSeries)

## Not run:
   # A multivariate zoo object:
   X.zoo = zoo(X.mat, order.by = as.Date(rownames(x.timeSeries)))
   garchFit(GARCH11 ~ garch(1,1), data = X.zoo)

## End(Not run)

   # A multivariate "mts" object:
   X.mts = as.ts(X.mat)
   garchFit(GARCH11 ~ garch(1,1), data = X.mts)

## MODELING THE PERCENTUAL SPI/SBI SPREAD FROM LPP BENCHMARK:

   stopifnot(require("timeSeries"))
   X.timeSeries = as.timeSeries(data(LPP2005REC))
   X.mat = as.matrix(X.timeSeries)
   ## Not run: X.zoo = zoo(X.mat, order.by = as.Date(rownames(X.mat)))
   X.mts = ts(X.mat)
   garchFit(100*(SPI - SBI) ~ garch(1,1), data = X.timeSeries)
   # The remaining are not yet supported ...
   # garchFit(100*(SPI - SBI) ~ garch(1,1), data = X.mat)
   # garchFit(100*(SPI - SBI) ~ garch(1,1), data = X.zoo)
   # garchFit(100*(SPI - SBI) ~ garch(1,1), data = X.mts)

## MODELING HIGH/LOW RETURN SPREADS FROM MSFT PRICE SERIES:

   X.timeSeries = MSFT
   garchFit(Open ~ garch(1,1), data = returns(X.timeSeries))
   garchFit(100*(High-Low) ~ garch(1,1), data = returns(X.timeSeries))

## GO-GARCH Modelling  (not yet!!) % FIXME

  ## data(DowJones30, package="fEcofin") # no longer exists
  ## X = returns(as.timeSeries(DowJones30)); head(X)
  ## N = 5; ans = .gogarchFit(data = X[, 1:N], trace = FALSE); ans
  ## ans@h.t
```

`garchFitControl` *Control GARCH fitting algorithms*

Description

Control parameters for the GARCH fitting algorithms.

Usage

```
garchFitControl(
    llh = c("filter", "internal", "testing"),
    nlminb.eval.max = 2000,
    nlminb.iter.max = 1500,
    nlminb.abs.tol = 1.0e-20,
    nlminb.rel.tol = 1.0e-14,
    nlminb.x.tol = 1.0e-14,
    nlminb.step.min = 2.2e-14,
    nlminb.scale = 1,
    nlminb.fscale = FALSE,
    nlminb.xscale = FALSE,
    sqp.mit = 200,
    sqp.mfv = 500,
    sqp.met = 2,
    sqp.mec = 2,
    sqp.mer = 1,
    sqp.mes = 4,
    sqp.xmax = 1.0e3,
    sqp.tolx = 1.0e-16,
    sqp.tolc = 1.0e-6,
    sqp.tolg = 1.0e-6,
    sqp.told = 1.0e-6,
    sqp.tols = 1.0e-4,
    sqp.rpf = 1.0e-4,
    lbfgsb.REPORT = 10,
    lbfgsb.lmm = 20,
    lbfgsb.pgtol = 1e-14,
    lbfgsb.factr = 1,
    lbfgsb.fnscale = FALSE,
    lbfgsb.parscale = FALSE,
    nm.ndeps = 1e-14,
    nm.maxit = 10000,
    nm.abstol = 1e-14,
    nm.reltol = 1e-14,
    nm.alpha = 1.0,
    nm.beta = 0.5,
    nm.gamma = 2.0,
    nm.fnscale = FALSE,
    nm.parscale = FALSE)
```

Arguments

`llh`	`llh = c("filter", "internal", "testing")[1]`, defaults to `"filter"`.
`nlminb.eval.max`	maximum number of evaluations of the objective function, defaults to 200.
`nlminb.iter.max`	maximum number of iterations, defaults to 150.

`nlminb.abs.tol` absolute tolerance, defaults to 1e-20.

`nlminb.rel.tol` relative tolerance, defaults to 1e-10.

`nlminb.x.tol` X tolerance, defaults to 1.5e-8.

`nlminb.fscale` defaults to FALSE.

`nlminb.xscale` defaulkts to FALSE.

`nlminb.step.min`
minimum step size, defaults to 2.2e-14.

`nlminb.scale` defaults to 1.

`sqp.mit` maximum number of iterations, defaults to 200.

`sqp.mfv` maximum number of function evaluations, defaults to 500.

`sqp.met` specifies scaling strategy:
sqp.met=1 - no scaling,
sqp.met=2 - preliminary scaling in 1st iteration (default),
sqp.met=3 - controlled scaling,
sqp.met=4 - interval scaling,
sqp.met=5 - permanent scaling in all iterations.

`sqp.mec` correction for negative curvature:
sqp.mec=1 - no correction,
sqp.mec=2 - Powell correction (default).

`sqp.mer` restarts after unsuccessful variable metric updates:
sqp.mer=0 - no restarts,
sqp.mer=1 - standard restart.

`sqp.mes` interpolation method selection in a line search:
sqp.mes=1 - bisection,
sqp.mes=2 - two point quadratic interpolation,
sqp.mes=3 - three point quadratic interpolation,
sqp.mes=4 - three point cubic interpolation (default).

`sqp.xmax` maximum stepsize, defaults to 1.0e+3.

`sqp.tolx` tolerance for the change of the coordinate vector, defaults to 1.0e-16.

`sqp.tolc` tolerance for the constraint violation, defaults to 1.0e-6.

`sqp.tolg` tolerance for the Lagrangian function gradient, defaults to 1.0e-6.

`sqp.told` defaults to 1.0e-6.

`sqp.tols` defaults to 1.0e-4.

`sqp.rpf` value of the penalty coefficient, default to1.0D-4. The default velue may be relatively small. Therefore, larger value, say one, can sometimes be more suitable.

`lbfgsb.REPORT` the frequency of reports for the `"BFGS"` and `"L-BFGS-B"` methods if `control$trace` is positive. Defaults to every 10 iterations.

`lbfgsb.lmm` an integer giving the number of BFGS updates retained in the `"L-BFGS-B"` method, It defaults to 5.

`lbfgsb.factr` controls the convergence of the `"L-BFGS-B"` method. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1.0e-8.

`lbfgsb.pgtol` helps control the convergence of the "L-BFGS-B" method. It is a tolerance on the projected gradient in the current search direction. This defaults to zero, when the check is suppressed.

`lbfgsb.fnscale` defaults to FALSE.

`lbfgsb.parscale`
defaults to FALSE.

`nm.ndeps` a vector of step sizes for the finite-difference approximation to the gradient, on par/parscale scale. Defaults to 1e-3.

`nm.maxit` the maximum number of iterations. Defaults to 100 for the derivative-based methods, and 500 for "Nelder-Mead". For "SANN" maxit gives the total number of function evaluations. There is no other stopping criterion. Defaults to 10000.

`nm.abstol` the absolute convergence tolerance. Only useful for non-negative functions, as a tolerance for reaching zero.

`nm.reltol` relative convergence tolerance. The algorithm stops if it is unable to reduce the value by a factor of `reltol * (abs(val) + reltol)` at a step. Defaults to `sqrt(.Machine$double.eps)`, typically about 1e-8.

`nm.alpha, nm.beta, nm.gamma`
scaling parameters for the "Nelder-Mead" method. alpha is the reflection factor (default 1.0), beta the contraction factor (0.5), and gamma the expansion factor (2.0).

`nm.fnscale` an overall scaling to be applied to the value of fn and gr during optimization. If negative, turns the problem into a maximization problem. Optimization is performed on `fn(par) / nm.fnscale`.

`nm.parscale` a vector of scaling values for the parameters. Optimization is performed on par/parscale and these should be comparable in the sense that a unit change in any element produces about a unit change in the scaled value.

Value

a list

Author(s)

Diethelm Wuertz for the Rmetrics R-port,
R Core Team for the 'optim' R-port,
Douglas Bates and Deepayan Sarkar for the 'nlminb' R-port,
Bell-Labs for the underlying PORT Library,
Ladislav Luksan for the underlying Fortran SQP Routine,
Zhu, Byrd, Lu-Chen and Nocedal for the underlying L-BFGS-B Routine.

See Also

garchFit

Examples

```
##
```

garchSim *Simulate univariate GARCH/APARCH time series*

Description

Simulates univariate GARCH/APARCH time series.

Usage

```
garchSim(spec = garchSpec(), n = 100, n.start = 100, extended = FALSE)
```

Arguments

spec	a specification object of class "fGARCHSPEC" as returned by garchSpec. See also below for further details.
n	length of the output series, an integer value, by default n=100.
n.start	length of 'burn-in' period, by default 100.
extended	logical parameter specifying what to return. If FALSE, return the univariate GARCH/APARCH time series. If TRUE, return a multivariate time series containing also the volatility and conditional innovations time series.

Details

The function garchSim simulates an univariate GARCH or APARCH time series process as specified by argument spec. The default model specifies Bollerslev's GARCH(1,1) model with normally distributed innovations.

spec is an object of class "fGARCHSPEC" as returned by the function garchSpec. It comes with a slot @model which is a list of just the numeric parameter entries. These are recognized and extracted for use by the function garchSim.

One can estimate the parameters of a GARCH process from empirical data using the function garchFit and then simulate statistically equivalent GARCH processes with the same set of model parameters using the function garchSim.

Value

the simulated time series as an objects of class "timeSeries" with attribute "spec" containing the specification of the model.

If extended is TRUE, then the time series is multivariate and contains also the volatility, sigma, and the conditional innovations, eps.

Note

An undocumented feature (so, it should not be relied on) is that the returned time series is timed so that the last observation is the day before the date when the function is executed. This probably should be controlled by an additional argument in garchSim.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

garchSpec, garchFit

Examples

```
## garchSpec -
   spec = garchSpec()
   spec

## garchSim -
   # Simulate a "timeSeries" object:
   x = garchSim(spec, n = 50)
   class(x)
   print(x)

## More simulations ...

   # Default GARCH(1,1) - uses default parameter settings
   spec = garchSpec(model = list())
   garchSim(spec, n = 10)

   # ARCH(2) - use default omega and specify alpha, set beta=0!
   spec = garchSpec(model = list(alpha = c(0.2, 0.4), beta = 0))
   garchSim(spec, n = 10)

   # AR(1)-ARCH(2) - use default mu, omega
   spec = garchSpec(model = list(ar = 0.5, alpha = c(0.3, 0.4), beta = 0))
   garchSim(spec, n = 10)

   # AR([1,5])-GARCH(1,1) - use default garch values and subset ar[.]
   spec = garchSpec(model = list(mu = 0.001, ar = c(0.5,0,0,0,0.1)))
   garchSim(spec, n = 10)

   # ARMA(1,2)-GARCH(1,1) - use default garch values
   spec = garchSpec(model = list(ar = 0.5, ma = c(0.3, -0.3)))
   garchSim(spec, n = 10)

   # GARCH(1,1) - use default omega and specify alpha/beta
   spec = garchSpec(model = list(alpha = 0.2, beta = 0.7))
   garchSim(spec, n = 10)

   # GARCH(1,1) - specify omega/alpha/beta
   spec = garchSpec(model = list(omega = 1e-6, alpha = 0.1, beta = 0.8))
   garchSim(spec, n = 10)

   # GARCH(1,2) - use default omega and specify alpha[1]/beta[2]
   spec = garchSpec(model = list(alpha = 0.1, beta = c(0.4, 0.4)))
   garchSim(spec, n = 10)
```

```
# GARCH(2,1) - use default omega and specify alpha[2]/beta[1]
spec = garchSpec(model = list(alpha = c(0.12, 0.04), beta = 0.08))
garchSim(spec, n = 10)

# snorm-ARCH(1) - use defaults with skew Normal
spec = garchSpec(model = list(beta = 0, skew = 0.8), cond.dist = "snorm")
garchSim(spec, n = 10)

# sged-GARCH(1,1) - using defaults with skew GED
model = garchSpec(model = list(skew = 0.93, shape = 3), cond.dist = "sged")
garchSim(model, n = 10)

# Taylor Schwert GARCH(1,1) - this belongs to the family of APARCH Models
spec = garchSpec(model = list(delta = 1))
garchSim(spec, n = 10)

# AR(1)-t-APARCH(2, 1) - a little bit more complex specification ...
spec = garchSpec(model = list(mu = 1.0e-4, ar = 0.5, omega = 1.0e-6,
    alpha = c(0.10, 0.05), gamma = c(0, 0), beta = 0.8, delta = 1.8,
    shape = 4, skew = 0.85), cond.dist = "sstd")
garchSim(spec, n = 10)

garchSim(spec, n = 10, extended = TRUE)
```

---

garchSpec *Univariate GARCH/APARCH time series specification*

Description

Specifies an univariate ARMA-GARCH or ARMA-APARCH time series model.

Usage

```
garchSpec(model = list(), presample = NULL,
    cond.dist = c("norm", "ged", "std", "snorm", "sged", "sstd"),
    rseed = NULL)
```

Arguments

cond.dist	a character string naming the desired conditional distribution. Valid values are "norm", "ged", "std", "snorm", "sged", "sstd". The default value is "norm", the standard normal distribution.
model	a list of GARCH model parameters, see section 'Details'. The default model=list() specifies Bollerslev's GARCH(1,1) model with normal conditional distributed innovations.

`presample` a numeric three column matrix with start values for the series, for the innovations, and for the conditional variances. For an ARMA(m,n)-GARCH(p,q) process the number of rows must be at least max(m,n,p,q)+1, longer presamples are truncated. Note, all presamples are initialized by a normal-GARCH(p,q) process.

`rseed` single integer argument, the seed for the intitialization of the random number generator for the innovations. If `rseed=NULL`, the default, then the state of the random number generator is not touched by this function.

Details

The function `garchSpec` specifies a GARCH or APARCH time series process which we can use for simulating artificial GARCH and/or APARCH models. This is very useful for testing the GARCH parameter estimation results, since your model parameters are known and well specified.

Argument `model` is a list of model parameters. For the GARCH part of the model they are:

`omega` the constant coefficient of the variance equation, by default `1e-6`;

`alpha` the value or vector of autoregressive coefficients, by default 0.1, specifying a model of order 1;

`beta` the value or vector of variance coefficients, by default 0.8, specifying a model of order 1.

If the model is APARCH, then the following additional parameters are available:

delta a positive number, the power of sigma in the volatility equation, it is 2 for GARCH models;

gamma the leverage parameters, a vector of length `alpha`, containing numbers in the interval $(0, 1)$.

The values for the linear part (conditional mean) are:

`mu` the mean value, by default NULL;

`ar` the autoregressive ARMA coefficients, by default NULL;

`ma` the moving average ARMA coefficients, by default NULL.

The parameters for the conditional distributions are:

`skew` the skewness parameter (also named "xi"), by default 0.9, effective only for the `"dsnorm"`, the `"dsged"`, and the `"dsstd"` skewed conditional distributions;

`shape` the shape parameter (also named "nu"), by default 2 for the `"dged"` and `"dsged"`, and by default 4 for the `"dstd"` and `"dsstd"` conditional distributions.

For example, specifying a subset AR(5[1,5])-GARCH(2,1) model with a standardized Student-t distribution with four degrees of freedom will return the following printed output:

```
garchSpec(model = list(ar = c(0.5,0,0,0,0.1), alpha =
    c(0.1, 0.1), beta = 0.75, shape = 4), cond.dist = "std")

Formula:
 ~ ar(5) + garch(2, 1)
```

```
Model:
 ar:    0.5 0 0 0 0.1
 omega: 1e-06
 alpha: 0.1 0.1
 beta:  0.75
Distribution:
 std
Distributional Parameter:
 nu = 4
Presample:
   time          z     h y
0     0 -0.3262334 2e-05 0
-1   -1  1.3297993 2e-05 0
-2   -2  1.2724293 2e-05 0
-3   -3  0.4146414 2e-05 0
-4   -4 -1.5399500 2e-05 0
```

Its interpretation is as follows. 'Formula' describes the formula expression specifying the generating process, 'Model' lists the associated model parameters, 'Distribution' the type of the conditional distribution function in use, 'Distributional Parameters' lists the distributional parameter (if any), and the 'Presample' shows the presample input matrix.

If we have specified `presample = NULL` in the argument list, then the presample is generated automatically by default as norm-AR()-GARCH() process.

Value

an object of class `"fGARCHSPEC"`

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

`garchSim`, `garchFit`

Examples

```
## garchSpec -

   # Normal Conditional Distribution:
   spec = garchSpec()
   spec

   # Skewed Normal Conditional Distribution:
   spec = garchSpec(model = list(skew = 0.8), cond.dist = "snorm")
   spec

   # Skewed GED Conditional Distribution:
```

```
spec = garchSpec(model = list(skew = 0.9, shape = 4.8), cond.dist = "sged")
spec

## More specifications ...

# Default GARCH(1,1) - uses default parameter settings
garchSpec(model = list())

# ARCH(2) - use default omega and specify alpha, set beta=0!
garchSpec(model = list(alpha = c(0.2, 0.4), beta = 0))

# AR(1)-ARCH(2) - use default mu, omega
garchSpec(model = list(ar = 0.5, alpha = c(0.3, 0.4), beta = 0))

# AR([1,5])-GARCH(1,1) - use default garch values and subset ar[.]
garchSpec(model = list(mu = 0.001, ar = c(0.5,0,0,0,0.1)))

# ARMA(1,2)-GARCH(1,1) - use default garch values
garchSpec(model = list(ar = 0.5, ma = c(0.3, -0.3)))

# GARCH(1,1) - use default omega and specify alpha/beta
garchSpec(model = list(alpha = 0.2, beta = 0.7))

# GARCH(1,1) - specify omega/alpha/beta
garchSpec(model = list(omega = 1e-6, alpha = 0.1, beta = 0.8))

# GARCH(1,2) - use default omega and specify alpha[1]/beta[2]
garchSpec(model = list(alpha = 0.1, beta = c(0.4, 0.4)))

# GARCH(2,1) - use default omega and specify alpha[2]/beta[1]
garchSpec(model = list(alpha = c(0.12, 0.04), beta = 0.08))

# snorm-ARCH(1) - use defaults with skew Normal
garchSpec(model = list(beta = 0, skew = 0.8), cond.dist = "snorm")

# sged-GARCH(1,1) - using defaults with skew GED
garchSpec(model = list(skew = 0.93, shape = 3), cond.dist = "sged")

# Taylor Schwert GARCH(1,1) - this belongs to the family of APARCH Models
garchSpec(model = list(delta = 1))

# AR(1)-t-APARCH(2, 1) - a little bit more complex specification ...
garchSpec(model = list(mu = 1.0e-4, ar = 0.5, omega = 1.0e-6,
    alpha = c(0.10, 0.05), gamma = c(0, 0), beta = 0.8, delta = 1.8,
    shape = 4, skew = 0.85), cond.dist = "sstd")
```

---

ged *Standardized generalized error distribution*

Description

Functions to compute density, distribution function, quantile function and to generate random variates for the standardized generalized error distribution.

Usage

```
dged(x, mean = 0, sd = 1, nu = 2, log = FALSE)
pged(q, mean = 0, sd = 1, nu = 2)
qged(p, mean = 0, sd = 1, nu = 2)
rged(n, mean = 0, sd = 1, nu = 2)
```

Arguments

x, q	a numeric vector of quantiles.
p	a numeric vector of probabilities.
n	number of observations to simulate.
mean	location parameter.
sd	scale parameter.
nu	shape parameter.
log	logical; if TRUE, densities are given as log densities.

Details

The standardized GED is defined so that for a given sd it has the same variance, sd^2, for all values of the shape parameter.

dstd computes the density, pstd the distribution function, qstd the quantile function, and rstd generates random deviates from the standardized-t distribution with the specified parameters.

Value

numeric vector

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Nelson D.B. (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

gedFit, absMoments, sged (skew GED),

gedSlider for visualization

Examples

```
## sged -
   par(mfrow = c(2, 2))
   set.seed(1953)
   r = rsged(n = 1000)
   plot(r, type = "l", main = "sged", col = "steelblue")

   # Plot empirical density and compare with true density:
   hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
   box()
   x = seq(min(r), max(r), length = 201)
   lines(x, dsged(x), lwd = 2)

   # Plot df and compare with true df:
   plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
     ylab = "Probability")
   lines(x, psged(x), lwd = 2)

   # Compute quantiles:
   round(qsged(psged(q = seq(-1, 5, by = 1))), digits = 6)
```

---

gedFit *Generalized error distribution parameter estimation*

Description

Function to fit the parameters of the generalized error distribution.

Usage

```
gedFit(x, ...)
```

Arguments

x	a numeric vector of quantiles.
...	parameters parsed to the optimization function nlm.

Value

gedFit returns a list with the following components:

par	The best set of parameters found.
objective	The value of objective corresponding to par.
convergence	An integer code, 0 indicates successful convergence.
message	A character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation.
iterations	Number of iterations performed.
evaluations	Number of objective function and gradient function evaluations.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Nelson D.B. (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

ged, sgedFit

Examples

```
## rged -
   set.seed(1953)
   r = rged(n = 1000)

## gedFit -
   gedFit(r)
```

---

gedSlider *Generalized error distribution slider*

Description

Displays interactively the dependence of the GED distribution on its parameters.

Usage

```
gedSlider(type = c("dist", "rand"))
```

Arguments

type	a character string denoting which interactive plot should be displayed. Either a distribution plot type = "dist", the default value, or a random variates plot, type = "rand".

Value

a Tcl object

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Nelson D.B. (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

ged, gedFit

Examples

```
## Not run:
## gedSlider -
   require(tcltk)
   gedSlider("dist")
   gedSlider("rand")

## End(Not run)
```

`plot-methods` *GARCH plot methods*

Description

Plot methods for GARCH Modelling.

Usage

```
## S4 method for signature 'fGARCH,missing'
plot(x, which = "ask", ...)
```

Arguments

`x`	an object of class `"fGARCH"`.
`which`	a character string or a vector of positive denoting which plot should be displayed, see section 'Details'.
`...`	optional arguments to be passed.

Details

The `plot` method for `"fGARCH"` objects offers a selection of diagnostic, exploratory, and presentation plots from a menu. Argument `which` can be used to request specific plots, which is particularly useful in scripts.

The generic function `plot` allows to display 13 graphs:

Time SeriesPlot

Conditional Standard Deviation Plot
Series Plot with 2 Conditional SD Superimposed
Autocorrelation function Plot of Observations
Autocorrelation function Plot of Squared Observations
Cross Correlation Plot
Residuals Plot
Conditional Standard Deviations Plot
Standardized Residuals Plot
ACF Plot of Standardized Residuals
ACF Plot of Squared Standardized Residuals
Cross Correlation Plot between r^2 and r
Quantile-Quantile Plot of Standardized Residuals

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

garchFit, class fGARCH,

predict, fitted, residuals

plot

Examples

```
## garchSim -
   # Default Garch(1,1) Model:
   x = garchSim(n = 200)
   head(x)

## garchFit -
   fit = garchFit(formula = ~ garch(1, 1), data = x, trace = FALSE)

## Batch Plot:
   plot(fit, which = 3)

## Not run:
## Plot:
   # Interactive Plot:
   plot(fit)

## End(Not run)
```

---

predict-methods *GARCH prediction function*

Description

Predicts a time series from a fitted GARCH object.

Usage

```
## S4 method for signature 'fGARCH'
predict(object, n.ahead = 10, trace = FALSE, mse = c("cond","uncond"),
        plot=FALSE, nx=NULL, crit_val=NULL, conf=NULL, ...)
```

Arguments

n.ahead	an integer value, denoting the number of steps to be forecasted, by default 10.
object	an object of class "fGARCH" as returned by the function garchFit.
trace	a logical flag. Should the prediction process be traced? By default trace=FALSE.
mse	If set to "cond", meanError is defined as the conditional mean errors $\sqrt{E_t[x_{t+h} - E_t(x_{t+h})]^2}$. If set to "uncond", it is defined as $\sqrt{E[x_{t+h} - E_t(x_{t+h})]^2}$.
plot	If set to TRUE, the confidence intervals are computed and plotted
nx	The number of observations to be plotted along with the predictions. The default is round(n*0.25), where n is the sample size.
crit_val	The critical values for the confidence intervals when plot is set to TRUE. The intervals are defined as $\hat{x}_{t+h}$ + crit_val[2] * meanError and $\hat{x}_{t+h}$ + crit_val[1] * meanError if two critical values are provided and $\hat{x}_{t+h} \pm$ crit_val * meanError if only one is given. If you do not provide critical values, they will be computed automatically.
conf	The confidence level for the confidence intervals if crit_val is not provided. By default it is set to 0.95. The critical values are then computed using the conditional distribution that was chosen to create the object with garchFit using the same shape and skew parameters. If the conditionnal distribution was set to "QMLE", the critical values are computed using the empirical distribution of the standardized residuals.
...	additional arguments to be passed.

Details

The predictions are returned as a data frame with with columns "meanForecast", "meanError", and "standardDeviation". Row h contains the predictions for horizon h.

The number of records equals the number of forecasting steps n.ahead.

Value

a data frame containing 3 columns and n.ahead rows, see section 'Details'

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

predict in base R

fitted, residuals,

plot, garchFit, class fGARCH,

Examples

```
## garchFit -
   # Parameter Estimation of Default GARCH(1,1) Model:
   set.seed(123)
   fit = garchFit(~ garch(1, 1), data = garchSim(), trace = FALSE)
   fit

## predict -
   predict(fit, n.ahead = 10)
   predict(fit, n.ahead = 10, mse="uncond")

## predict with plotting: critical values = +/- 2

   predict(fit, n.ahead = 10, plot=TRUE, crit_val = 2)

## predict with plotting: automatic critical values
## for different conditional distributions

  set.seed(321)
  fit2 = garchFit(~ garch(1, 1), data = garchSim(), trace=FALSE, cond.dist="sged")

## 95% confidence level
 predict(fit2, n.ahead=20, plot=TRUE)

 set.seed(444)
 fit3 = garchFit(~ garch(1, 1), data = garchSim(), trace=FALSE, cond.dist="QMLE")

## 90% confidence level and nx=100 :
 predict(fit3, n.ahead=20, plot=TRUE, conf=.9, nx=100)
```

`residuals-methods` *Extract GARCH model residuals*

Description

Extracts residuals from a fitted GARCH object.

Usage

```
## S4 method for signature 'fGARCH'
residuals(object, standardize = FALSE)
```

Arguments

`object`	an object of class `"fGARCH"` as returned from `garchFit`.
`standardize`	a logical, indicating if the residuals should be standardized.

Details

residuals is a generic function which extracts residual values from objects returned by modeling functions.

The "fGARCH" method extracts the @residuals slot from an object of class "fGARCH" as returned by the function garchFit and optionally standardizes the residuals, using conditional standard deviations.

The class of the returned value depends on the input to the function garchFit who created the object. The returned value is always of the same class as the input object to the argument data in the function garchFit, i.e. if you fit a "timeSeries" object, you will get back from the function fitted also a "timeSeries" object, if you fit an object of class "zoo", you will get back again a "zoo" object. The same holds for a "numeric" vector, for a "data.frame", and for objects of class "ts", "mts".

In contrast, the slot itself returns independent of the class of the data input always a numeric vector, i.e. the function call rslot(object, "fitted") will return a numeric vector.

Note

(GNB) Contrary to the description of the returned value of the "fGARCH" method, it is always "numeric".

TODO: either implement the documented behaviour or fix the documentation.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

fitted, predict, garchFit, class fGARCH,

Examples

```
## Swiss Pension fund Index -
   stopifnot(require("timeSeries"))
   x = as.timeSeries(data(LPP2005REC))

## garchFit
   fit = garchFit(LPP40 ~ garch(1, 1), data = 100*x, trace = FALSE)
   fit

## residuals -
   res = residuals(fit)
   head(res)
   class(res)

## slot -
   identical(res, slot(fit, "residuals"))
```

sged *Skew generalized error distribution*

Description

Functions to compute density, distribution function, quantile function and to generate random variates for the skew generalized error distribution.

Usage

```
dsged(x, mean = 0, sd = 1, nu = 2, xi = 1.5, log = FALSE)
psged(q, mean = 0, sd = 1, nu = 2, xi = 1.5)
qsged(p, mean = 0, sd = 1, nu = 2, xi = 1.5)
rsged(n, mean = 0, sd = 1, nu = 2, xi = 1.5)
```

Arguments

`mean, sd, nu, xi`	location parameter `mean`, scale parameter `sd`, shape parameter `nu`, skewness parameter `xi`.
`n`	the number of observations.
`p`	a numeric vector of probabilities.
`x, q`	a numeric vector of quantiles.
`log`	a logical; if TRUE, densities are given as log densities.

Value

`d*` returns the density, `p*` returns the distribution function, `q*` returns the quantile function, and `r*` generates random deviates,
all values are numeric vectors.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Nelson D.B. (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

`sgedFit` (fit), `sgedSlider` (visualize),

`ged` (symmetric GED)

Examples

```
## sged -
   par(mfrow = c(2, 2))
   set.seed(1953)
   r = rsged(n = 1000)
   plot(r, type = "l", main = "sged", col = "steelblue")

   # Plot empirical density and compare with true density:
   hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
   box()
   x = seq(min(r), max(r), length = 201)
   lines(x, dsged(x), lwd = 2)

   # Plot df and compare with true df:
   plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
     ylab = "Probability")
   lines(x, psged(x), lwd = 2)

   # Compute quantiles:
   round(qsged(psged(q = seq(-1, 5, by = 1))), digits = 6)
```

---

sgedFit *Skew generalized error distribution parameter estimation*

Description

Function to fit the parameters of the skew generalized error distribution.

Usage

```
sgedFit(x, ...)
```

Arguments

x	a numeric vector of quantiles.
...	parameters parsed to the optimization function nlm.

Value

sgedFit returns a list with the following components:

par	The best set of parameters found.
objective	The value of objective corresponding to par.
convergence	An integer code. 0 indicates successful convergence.
message	A character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation.
iterations	Number of iterations performed.
evaluations	Number of objective function and gradient function evaluations.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Nelson D.B. (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

sged, sgedSlider

Examples

```
## rsged -
   set.seed(1953)
   r = rsged(n = 1000)

## sgedFit -
   sgedFit(r)
```

---

sgedSlider *Skew GED distribution slider*

Description

Displays interactively the dependence of the skew GED distribution on its parameters.

Usage

```
sgedSlider(type = c("dist", "rand"))
```

Arguments

type	a character string denoting which interactive plot should be displayed. Either a distribution plot `type="dist"`, the default value, or a random variates plot, `type="rand"`.

Value

a Tcl object

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Nelson D.B. (1991); *Conditional Heteroscedasticity in Asset Returns: A New Approach*, Econometrica, 59, 347–370.

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

sged, sgedFit

Examples

```
## Not run:
## sgedSlider -
   require(tcltk)
   sgedSlider("dist")
   sgedSlider("rand")

## End(Not run)
```

---

snorm *Skew normal distribution*

Description

Functions to compute density, distribution function, quantile function and to generate random variates for the skew normal distribution.

Usage

```
dsnorm(x, mean = 0, sd = 1, xi = 1.5, log = FALSE)
psnorm(q, mean = 0, sd = 1, xi = 1.5)
qsnorm(p, mean = 0, sd = 1, xi = 1.5)
rsnorm(n, mean = 0, sd = 1, xi = 1.5)
```

Arguments

x, q	a numeric vector of quantiles.
p	a numeric vector of probabilities.
n	the number of observations.
mean	location parameter.
sd	scale parameter.
xi	skewness parameter.
log	a logical; if TRUE, densities are given as log densities.

Details

dsnorm computed the density, psnorm the distribution function, qsnorm the quantile function, and rsnorm generates random deviates.

Value

numeric vector

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

snormFit (fit), snormSlider (visualize),

sstd (skew Student-t), sged (skew GED)

Examples

```
## snorm -
   # Ranbdom Numbers:
   par(mfrow = c(2, 2))
   set.seed(1953)
   r = rsnorm(n = 1000)
   plot(r, type = "l", main = "snorm", col = "steelblue")

   # Plot empirical density and compare with true density:
   hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
   box()
   x = seq(min(r), max(r), length = 201)
   lines(x, dsnorm(x), lwd = 2)

   # Plot df and compare with true df:
   plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
     ylab = "Probability")
   lines(x, psnorm(x), lwd = 2)

   # Compute quantiles:
   round(qsnorm(psnorm(q = seq(-1, 5, by = 1))), digits = 6)
```

snormFit *Skew normal distribution parameter estimation*

Description

Fits the parameters of the skew normal distribution.

Usage

```
snormFit(x, ...)
```

Arguments

x	a numeric vector of quantiles.
...	parameters passed to the optimization function nlm.

Value

snormFit returns a list with the following components:

par	The best set of parameters found.
objective	The value of objective corresponding to par.
convergence	An integer code. 0 indicates successful convergence.
message	A character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation.
iterations	Number of iterations performed.
evaluations	Number of objective function and gradient function evaluations.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

snormFit (fit), snormSlider (visualize), absMoments

Examples

```
## rsnorm -
   set.seed(1953)
   r = rsnorm(n = 1000)

## snormFit -
   snormFit(r)
```

`snormSlider` *Skew normal distribution slider*

Description

Displays interactively the dependence of the skew Normal distribution on its parameters.

Usage

```
snormSlider(type = c("dist", "rand"))
```

Arguments

`type`	a character string denoting which interactive plot should be displayed. Either a distribution plot `type="dist"`, the default value, or a random variates plot, `type="rand"`.

Value

a Tcl object

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

`snormFit` (fit), `snorm`,

Examples

```
## Not run:
## snormSlider -
   require(tcltk)
   snormSlider("dist")
   snormSlider("rand")

## End(Not run)
```

`sstd` *Skew Student-t distribution*

Description

Functions to compute density, distribution function, quantile function and to generate random variates for the skew Student-t distribution.

Usage

```
dsstd(x, mean = 0, sd = 1, nu = 5, xi = 1.5, log = FALSE)
psstd(q, mean = 0, sd = 1, nu = 5, xi = 1.5)
qsstd(p, mean = 0, sd = 1, nu = 5, xi = 1.5)
rsstd(n, mean = 0, sd = 1, nu = 5, xi = 1.5)
```

Arguments

`x, q`	a numeric vector of quantiles.
`p`	a numeric vector of probabilities.
`n`	number of observations to simulate.
`mean`	location parameter.
`sd`	scale parameter.
`nu`	shape parameter (degrees of freedom).
`xi`	skewness parameter.
`log`	logical; if `TRUE`, densities are given as log densities.

Details

`dsstd` computes the density, `psstd` the distribution function, `qsstd` the quantile function, and `rsstd` generates random deviates.

Value

numeric vector

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

`sstdFit` (fit), `sstdSlider` (visualize)

Examples

```
## sstd -
   par(mfrow = c(2, 2))
   set.seed(1953)
   r = rsstd(n = 1000)
   plot(r, type = "l", main = "sstd", col = "steelblue")

   # Plot empirical density and compare with true density:
   hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
   box()
   x = seq(min(r), max(r), length = 201)
   lines(x, dsstd(x), lwd = 2)

   # Plot df and compare with true df:
   plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
     ylab = "Probability")
   lines(x, psstd(x), lwd = 2)

   # Compute quantiles:
   round(qsstd(psstd(q = seq(-1, 5, by = 1))), digits = 6)
```

---

sstdFit *Skew Student-t distribution parameter estimation*

Description

Fits the parameters of the skew Student-t distribution.

Usage

```
sstdFit(x, ...)
```

Arguments

`x`	a numeric vector of quantiles.
`...`	parameters passed to the optimization function `nlm`.

Value

sstdFit returns a list with the following components:

par	The best set of parameters found.
objective	The value of objective corresponding to par.
convergence	An integer code. 0 indicates successful convergence.
message	A character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation.
iterations	Number of iterations performed.
evaluations	Number of objective function and gradient function evaluations.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

sstd, stdFit

Examples

```
## sstd -
   set.seed(1953)
   r = rsstd(n = 1000)


## sstdFit -
   sstdFit(r)
```

---

sstdSlider *Skew Student-t distribution slider*

Description

Displays interactively the dependence of the skew Student-t distribution on its parameters.

Usage

```
sstdSlider(type = c("dist", "rand"))
```

Arguments

type	a character string denoting which interactive plot should be displayed. Either a distribution plot type="dist", the default value, or a random variates plot, type="rand".

Value

a Tcl object

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

sstd, sstdFit

Examples

```
## Not run:
## sstdSlider -
   require(tcltk)
   sstdSlider("dist")
   sstdSlider("rand")

## End(Not run)
```

---

std *Standardized Student-t distribution*

Description

Functions to compute density, distribution function, quantile function and to generate random variates for the standardized Student-t distribution.

Usage

```
dstd(x, mean = 0, sd = 1, nu = 5, log = FALSE)
pstd(q, mean = 0, sd = 1, nu = 5)
qstd(p, mean = 0, sd = 1, nu = 5)
rstd(n, mean = 0, sd = 1, nu = 5)
```

Arguments

x, q	a numeric vector of quantiles.
p	a numeric vector of probabilities.
n	number of observations to simulate.
mean	location parameter.
sd	scale parameter.
nu	shape parameter (degrees of freedom).
log	logical; if TRUE, densities are given as log densities.

Details

The standardized Student-t distribution is defined so that for a given sd it has the same variance, sd^2, for all degrees of freedom. For comparison, the variance of the usual Student-t distribution is nu/(nu-2), where nu is the degrees of freedom. The usual Student-t distribution is obtained by setting sd = sqrt(nu/(nu - 2)).

Argument nu must be greater than 2. Although there is a default value for nu, it is rather arbitrary and relying on it is strongly discouraged.

dstd computes the density, pstd the distribution function, qstd the quantile function, and rstd generates random deviates from the standardized-t distribution with the specified parameters.

Value

numeric vector

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

stdFit (fit). stdSlider (visualize),

absMoments

Examples

```
## std -

pstd(1, sd = sqrt(5/(5-2)), nu = 5) == pt(1, df = 5) # TRUE

   par(mfrow = c(2, 2))
   set.seed(1953)
   r = rstd(n = 1000)
```

```
plot(r, type = "l", main = "sstd", col = "steelblue")

# Plot empirical density and compare with true density:
hist(r, n = 25, probability = TRUE, border = "white", col = "steelblue")
box()
x = seq(min(r), max(r), length = 201)
lines(x, dstd(x), lwd = 2)

# Plot df and compare with true df:
plot(sort(r), (1:1000/1000), main = "Probability", col = "steelblue",
  ylab = "Probability")
lines(x, pstd(x), lwd = 2)

# Compute quantiles:
round(qstd(pstd(q = seq(-1, 5, by = 1))), digits = 6)
```

---

stdFit *Student-t distribution parameter estimation*

Description

Fits the parameters of the standardized Student-t distribution.

Usage

```
stdFit(x, ...)
```

Arguments

x	a numeric vector of quantiles.
...	parameters parsed to the optimization function nlm.

Value

stdFit returns a list with the following components:

par	The best set of parameters found.
objective	The value of objective corresponding to par.
convergence	An integer code. 0 indicates successful convergence.
message	A character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation.
iterations	Number of iterations performed.
evaluations	Number of objective function and gradient function evaluations.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

References

Fernandez C., Steel M.F.J. (2000); *On Bayesian Modelling of Fat Tails and Skewness*, Preprint, 31 pages.

See Also

std, stdSlider

Examples

```
## std -
   set.seed(1953)
   r = rstd(n = 1000)


## stdFit -
   stdFit(r)
```

---

stdSlider *Student-t distribution slider*

Description

Displays interactively the dependence of the Student-t distribution on its parameters.

Usage

```
stdSlider(type = c("dist", "rand"))
```

Arguments

type	a character string denoting which interactive plot should be displayed. Either a distribution plot `type="dist"`, the default value, or a random variates plot, `type="rand"`.

Value

a Tcl object

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

std, stdFit,

Examples

```
## Not run:
## stdSlider -
   require(tcltk)
   stdSlider("dist")
   stdSlider("rand")

## End(Not run)
```

---

summary-methods *GARCH summary methods*

Description

Summary methods for GARCH modelling.

Methods

Methods for summary defined in package **fGarch**:

object = "fGARCH" Summary function for objects of class "fGARCH".

How to read a diagnostic summary report?

The first five sections return the title, the call, the mean and variance formula, the conditional distribution and the type of standard errors:

```
Title:
 GARCH Modelling

Call:
 garchFit(~ garch(1, 1), data = garchSim(), trace = FALSE)

Mean and Variance Equation:
 ~arch(0)

Conditional Distribution:
 norm

Std. Errors:
 based on Hessian
```

The next three sections return the estimated coefficients and an error analysis including standard errors, t values, and probabilities, as well as the log Likelihood values from optimization:

```
Coefficient(s):
          mu         omega        alpha1         beta1
-5.79788e-05   7.93017e-06   1.59456e-01   2.30772e-01

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -5.798e-05   2.582e-04   -0.225    0.822
omega   7.930e-06   5.309e-06    1.494    0.135
alpha1  1.595e-01   1.026e-01    1.554    0.120
beta1   2.308e-01   4.203e-01    0.549    0.583

Log Likelihood:
 -843.3991    normalized:  -Inf
```

The next section provides results on standardized residuals tests, including statistic and p values, and on information criterion statistic including AIC, BIC, SIC, and HQIC:

```
Standardized Residuals Tests:
                               Statistic p-Value
 Jarque-Bera Test   R    Chi^2  0.4172129 0.8117146
 Shapiro-Wilk Test  R    W      0.9957817 0.8566985
 Ljung-Box Test     R    Q(10)  13.05581  0.2205680
 Ljung-Box Test     R    Q(15)  14.40879  0.4947788
 Ljung-Box Test     R    Q(20)  38.15456  0.008478302
 Ljung-Box Test     R^2  Q(10)  7.619134  0.6659837
 Ljung-Box Test     R^2  Q(15)  13.89721  0.5333388
 Ljung-Box Test     R^2  Q(20)  15.61716  0.7400728
 LM Arch Test       R    TR^2   7.049963  0.8542942

Information Criterion Statistics:
         AIC      BIC      SIC     HQIC
    8.473991 8.539957 8.473212 8.500687
```

Author(s)

Diethelm Wuertz for the Rmetrics R-port

Examples

```
## garchSim -
   x = garchSim(n = 200)

## garchFit -
   fit = garchFit(formula = x ~ garch(1, 1), data = x, trace = FALSE)
   summary(fit)
```

`volatility-methods` *Extract GARCH model volatility*

Description

Extracts volatility from a fitted GARCH object.

Usage

```
volatility(object, ...)
## S3 method for class 'fGARCH'
volatility(object, type = c("sigma", "h"), ...)
```

Arguments

`object`	an object of class `"fGARCH"` as returned by `garchFit()`.
`type`	a character string denoting if the conditional standard deviations `"sigma"` or the variances `"h"` should be returned.
`...`	additional arguments to be passed.

Details

`volatility` is an S3 generic function for computation of volatility, see `link[fBasics]{volatility}` for the default method.

The method for `"fGARCH"` objects, described here, extracts the volatility from slot `@sigma.t` or `@h.t` of an `"fGARCH"` object usually obtained from the function `garchFit()`.

The class of the returned value depends on the input to the function `garchFit` who created the object. The returned value is always of the same class as the input object to the argument `data` in the function `garchFit`, i.e. if you fit a `"timeSeries"` object, you will get back from the function `fitted` also a `"timeSeries"` object, if you fit an object of class `"zoo"`, you will get back again a `"zoo"` object. The same holds for a `"numeric"` vector, for a `"data.frame"`, and for objects of class `"ts"`, `"mts"`.

In contrast, the slot itself always contains a numeric vector, independently of the class of the input data input, i.e. the function call `slot(object, "fitted")` will return a numeric vector.

Methods

Methods for `volatility` defined in package **fGarch**:

object = "fGARCH" Extractor function for volatility or standard deviation from an object of class `"fGARCH"`.

Note

(GNB) Contrary to the description of the returned value of the `"fGARCH"` method, it is always `"numeric"`.

TODO: either implement the documented behaviour or fix the documentation.

Author(s)

Diethelm Wuertz for the Rmetrics R-port

See Also

garchFit, class fGARCH

Examples

```
## Swiss Pension fund Index -
   stopifnot(require("timeSeries")) # need package 'timeSeries'
   x = as.timeSeries(data(LPP2005REC, package = "timeSeries"))

## garchFit
   fit = garchFit(LPP40 ~ garch(1, 1), data = 100*x, trace = FALSE)
   fit

## volatility -
   # Standard Deviation:
   vola = volatility(fit, type = "sigma")
   head(vola)
   class(vola)
   # Variance:
   vola = volatility(fit, type = "h")
   head(vola)
   class(vola)

## slot -
   vola = slot(fit, "sigma.t")
   head(vola)
   class(vola)
   vola = slot(fit, "h.t")
   head(vola)
   class(vola)
```

Index