Package ‘betategarch’

October 12, 2022

Type Package

Title Simulation, Estimation and Forecasting of Beta-Skew-t-EGARCH Models

Version 3.3

Depends R (>= 2.15.0), zoo

Date 2016-10-16

Author Genaro Sucarrat

Maintainer Genaro Sucarrat <genaro.sucarrat@bi.no>

URL http://www.sucarrat.net/

Description Simulation, estimation and forecasting of first-order Beta-Skew-t-EGARCH models with leverage (one-component, two-component, skewed versions).

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-10-16 20:09:51

R topics documented:

betategarch-package 2
coef.tegarch 3
dST 6
nasdaq 7
predict.tegarch 8
tegarch 10
tegarchLogl 12
tegarchSim 14

Index **17**

betategarch-package *Simulation, estimation and forecasting of Beta-Skew-t-EGARCH models*

Description

This package provides facilities for the simulation, estimation and forecasting of first order Beta-Skew-t-EGARCH models with leverage (one-component and two-component versions), see Harvey and Sucarrat (2014), and Sucarrat (2013).

Let y[t] denote a financial return at time t equal to

y[t] = sigma[t]*epsilon[t]

where sigma[t] > 0 is the scale or volatility (generally not equal to the conditional standard deviation), and where epsilon[t] is IID and t-distributed (possibly skewed) with df degrees of freedom. Then the first order log-volatility specifiction of the one-component Beta-Skew-t-EGARCH model can be parametrised as

sigma[t] = exp(lambda[t]),
lambda[t] = omega + lambdadagger,
lambdadagger[t] = phi1*lambdadagger[t-1] + kappa1*u[t-1] + kappastar*sign[-y]*(u[t-1]+1).

So the scale or volatility is given by sigma[t] = exp(lambda[t]). The omega is the unconditional or long-term log-volatility, phi1 is the GARCH parameter (|phi1| < 1 implies stability), kappa1 is the ARCH parameter, kappastar is the leverage or volatility-asymmetry parameter and u[t] is the conditional score or first derivative of the log-likelihood with respect to lambda. The score u[t] is zero-mean and IID, and (u[t]+1)/(df+1) is Beta distributed when there is no skew in the conditional density of epsilon[t]. The two-component specification is given by

sigma[t] = exp(lambda[t]),
lambda[t] = omega + lambda1dagger + lambda2dagger,
lambda1dagger[t] = phi1*lambdadagger[t-1] + kappa1*u[t-1],
lambda2dagger[t] = phi2*lambdadagger[t-1] + kappa2*u[t-1] + kappastar*sign[-y]*(u[t-1]+1).

The first component, lambda1dagger, is interpreted as the long-term component, whereas the second component, lambda2dagger, is interpreted as the short-term component.

Details

Package:	betategarch
Type:	Package
Version:	3.3
Date:	2016-10-16
License:	GPL-2
LazyLoad:	yes

The two main functions of the package are `tegarchSim` and `tegarch`. The first simulates a Beta-Skew-t-EGARCH models whereas the second estimates one. The second object returns an object (a list) of class 'tegarch', and a collection of methods can be applied to this class: `coef.tegarch`, `fitted.tegarch`, `logLik.tegarch`, `predict.tegarch`, `print.tegarch`, `residuals.tegarch`, `summary.tegarch` and `vcov.tegarch`. In addition, the output produced by the `tegarchSim` function and the `fitted.tegarch` and `residuals.tegarch` methods are of the Z's ordered observations (`zoo`) class, which means a range of time-series methods are available for these objects.

Author(s)

Genaro Sucarrat, http://www.sucarrat.net/

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.

Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

Examples

```
#simulate 500 observations from model with default parameter values:
set.seed(123)
y <- tegarchSim(500)

#estimate and store as 'mymod':
mymod <- tegarch(y)

#print estimates and standard errors:
print(mymod)

#graph of fitted volatility (conditional standard deviation):
plot(fitted(mymod))

#plot forecasts of volatility 1-step ahead up to 10-steps ahead:
plot(predict(mymod, n.ahead=10))
```

`coef.tegarch` *Extraction methods for 'tegarch' objects*

Description

Extraction methods for objects of class 'tegarch' (i.e. the result of estimating a Beta-Skew-t-EGARCH model)

Usage

```
## S3 method for class 'tegarch'
coef(object, ...)
## S3 method for class 'tegarch'
fitted(object, verbose = FALSE, ...)
## S3 method for class 'tegarch'
logLik(object, ...)
## S3 method for class 'tegarch'
print(x, ...)
## S3 method for class 'tegarch'
residuals(object, standardised = TRUE, ...)
## S3 method for class 'tegarch'
summary(object, verbose = FALSE, ...)
## S3 method for class 'tegarch'
vcov(object, ...)
```

Arguments

`object`	an object of class 'tegarch'
`x`	an object of class 'tegarch'
`verbose`	logical. If FALSE (default) then only basic information is returned
`standardised`	logical. If TRUE (default) then the standardised residuals are returned. If FALSE then the scaled (by sigma) residuals are returned
`...`	additional arguments

Details

Empty

Value

`coef:`	A numeric vector containing the parameter estimates
`fitted:`	A zoo object. If verbose=FALSE (default), then the zoo object is a vector containing the fitted conditional standard deviations. If verbose = TRUE, then the zoo object is a matrix containing the return series y, fitted scale (sigma), fitted conditional standard deviation (stdev), fitted log-scale (lambda), dynamic component(s) (lambdadagger in the 1-component specification, lambda1dagger and lambda2dagger in the 2-compoment specification), the score (u), scaled residuals (epsilon) and standardised residuals (residstd)
`logLik:`	The value of the log-likelihood at the maximum
`print:`	Prints the most important parts of the estimation results
`residuals:`	A zoo object. If standardised = TRUE (default), then the zoo object is a vector with the standardised residuals. If standardised = FALSE, then the zoo vector contains the scaled residuals
`summary:`	A list. If verbose = FALSE, then only the most important entries are returned. If verbose = TRUE, then all entries apart from the 1st. (the y series) is returned

`vcov:` The variance-covariance matrix of the estimated coefficents. The matrix is obtained by inverting the numerically estimated Hessian

Author(s)

Genaro Sucarrat, http://www.sucarrat.net/

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.

Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

See Also

`tegarch`, `coef`, `fitted`, `logLik`, `predict`, `predict.tegarch`, `print`, `summary`, `vcov`

Examples

```
#simulate 500 observations from model with default parameter values:
set.seed(123)
y <- tegarchSim(500)

#estimate and store as 'mymodel':
mymod <- tegarch(y)

#print estimation result:
print(mymod)

#extract coefficients:
coef(mymod)

#extract log-likelihood:
logLik(mymod)

#plot fitted conditional standard deviations:
plot(fitted(mymod))

#plot all the fitted series:
plot(fitted(mymod, verbose=TRUE))

#histogram of standardised residuals:
hist(residuals(mymod))
```

dST *The skewed t distribution*

Description

Density, random number generation, mean, variance, skewness and kurtosis functions for the uncentred skewed t distribution. The skewing method is that of Fernandez and Steel (1998).

Usage

```
dST(y, df = 10, sd = 1, skew = 1, log = FALSE)
rST(n, df = 10, skew = 1)
STmean(df, skew = 1)
STvar(df, skew = 1)
STskewness(df, skew = 1)
STkurtosis(df, skew = 1)
```

Arguments

`y`	numeric vector of quantiles
`n`	integer, the number of observations
`df`	degrees of freedom, greater than 0 and less than Inf
`sd`	scale, greater than 0 and less than Inf
`skew`	skewness, greater than 0 and less than Inf. Symmetry obtains when skew = 1 (default).
`log`	logical. TRUE returns the natural log of the density value, FALSE (default) returns the density value.

Details

Empty

Value

`dST:`	a numeric value, either the density value or the natural log of the density value
`rST:`	a numeric vector with n random numbers
`STmean:`	The mean of an uncentred skewed t variable
`STvar:`	The variance of an uncentred skewed t variable
`STskewness:`	3rd. moment of a standardised skewed t variable
`STkurtosis:`	4th. moment of a standardised skewed t variable

Note

Empty

Author(s)

Genaro Sucarrat, http://www.sucarrat.net/

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

See Also

tegarchSim

Examples

```
##generate 1000 random numbers from the skewed t:
set.seed(123)
eps <- rST(500, df=5) #symmetric t
eps <- rST(500, df=5, skew=0.8) #skewed to the left
eps <- rST(500, df=5, skew=2) #skewed to the right

##compare empirical mean with analytical:
mean(eps)
STmean(5, skew=2)

##compare empirical variance with analytical:
var(eps)
STvar(5, skew=2)
```

---

nasdaq *Daily Apple stock returns*

Description

The dataset contains two variables, day and nasdaqret. Day is the date of the return and nasdaqret is the daily (closing value) log-return in percent of the Apple stock over the period 10 September 1985 - 10 May 2011 (a total of 6835 observations).

Usage

```
data(nasdaq)
```

Format

A data frame with 3215 observations:

`day` a factor

`nasdaqret` a numeric vector

Details

The data is studied in more detail in Harvey and Sucarrat (2014), and in Sucarrat (2013).

Source

The source of the original raw data is http://yahoo.finance.com/.

References

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.
Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

Examples

```
data(nasdaq) #load data into workspace
mymod <- tegarch(nasdaq[,"nasdaqret"]) #estimate volatility model of Apple returns
print(mymod)
```

---

predict.tegarch — *Generate volatility forecasts* n*-steps ahead*

Description

Generates volatility forecasts from a model fitted by `tegarch` (i.e. a Beta-Skew-t-EGARCH model)

Usage

```
## S3 method for class 'tegarch'
predict(object, n.ahead = 1, initial.values = NULL, n.sim = 10000,
  verbose = FALSE, ...)
```

Arguments

`object`	an object of class 'tegarch'.
`n.ahead`	the number of steps ahead for which prediction is required.
`initial.values`	a vector containing the initial values of lambda and lambdadagger (lambda1dagger and lambda2dagger for 2-component models). If NULL (default) then the fitted values associated with the last return-observation are used
`n.sim`	number of simulated skew t variates.
`verbose`	logical. If FALSE (default) then only the conditional standard deviations are returned. If TRUE then also the scale is returned.
`...`	additional arguments

Details

The forecast formulas of exponential ARCH models are much more complicated than those of ordinary or non-exponential ARCH models. This is particularly the case when the conditional density is skewed. The forecast formula of the conditional scale of the Beta-Skew-t-EGARCH model is not available in closed form. Accordingly, some terms (expectations involving the skewed t) are estimated numerically by means of simulation.

Value

A zoo object. If verbose = FALSE, then the zoo object is a vector with the forecasted conditional standard deviations. If verbose = TRUE, then the zoo object is a matrix with forecasts of both the conditional scale and the conditional standard deviation

Author(s)

Genaro Sucarrat, http://www.sucarrat.net/

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.

Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

See Also

tegarch, predict

Examples

```
##simulate series with 500 observations:
set.seed(123)
y <- tegarchSim(500, omega=0.01, phi1=0.9, kappa1=0.1, kappastar=0.05, df=10, skew=0.8)

##estimate a 1st. order Beta-t-EGARCH model and store the output in mymod:
mymod <- tegarch(y)

#plot forecasts of volatility 1-step ahead up to 10-steps ahead:
plot(predict(mymod, n.ahead=10))
```

`tegarch` *Estimate first order Beta-Skew-t-EGARCH models*

Description

Fits a first order Beta-Skew-t-EGARCH model to a univariate time-series by exact Maximum Likelihood (ML) estimation. Estimation is via the `nlminb` function

Usage

```
tegarch(y, asym = TRUE, skew = TRUE, components = 1, initial.values = NULL,
  lower = NULL, upper = NULL, hessian = TRUE, lambda.initial = NULL,
  c.code = TRUE, logl.penalty = NULL, aux = NULL, ...)
```

Arguments

`y`	numeric vector, typically a financial return series.
`asym`	logical. TRUE (default) includes leverage or volatility asymmetry in the log-scale specification
`skew`	logical. TRUE (default) enables and estimates the skewness in conditional density (epsilon). The skewness method is that of Fernandez and Steel (1998)
`components`	Numeric value, either 1 (default) or 2. The former estimates a 1-component model, the latter a 2-component model
`initial.values`	NULL (default) or a vector with the initial values. If NULL, then the values are automatically chosen according to model (with or without skewness, 1 or 2 components, etc.)
`lower`	NULL (default) or a vector with the lower bounds of the parameter space. If NULL, then the values are automatically chosen
`upper`	NULL (default) or a vector with the upper bounds of the parameter space. If NULL, then the values are automatically chosen
`hessian`	logical. If TRUE (default) then the Hessian is computed numerically via the optimHess function. Setting hessian=FALSE speeds up estimation, which might be particularly useful in simulation. However, it also slows down the extraction of the variance-covariance matrix by means of the vcov method.
`lambda.initial`	NULL (default) or a vector with the initial value(s) of the recursion for lambda and lambdadagger. If NULL then the values are chosen automatically
`c.code`	logical. TRUE (default) is faster since it makes use of compiled C-code
`logl.penalty`	NULL (default) or a numeric value. If NULL then the log-likelihood value associated with the initial values is used. Sometimes estimation can result in NA and/or +/-Inf values, which are fatal for simulations. The value logl.penalty is the value returned by the log-likelihood function in the presence of NA or +/-Inf values
`aux`	NULL (default) or a list, se code. Useful for simulations (speeds them up)
`...`	further arguments passed to the nlminb function

Value

Returns a list of class 'tegarch' with the following elements:

`y`	the series used for estimation.
`date`	date and time of estimation.
`initial.values`	initial values used in estimation.
`lower`	lower bounds used in estimation.
`upper`	upper bounds used in estimation.
`lambda.initial`	initial values of lambda provided by the user, if any.
`model`	type of model estimated.
`hessian`	the numerically estimated Hessian.
`sic`	the value of the Schwarz (1978) information criterion.
`par`	parameter estimates.
`objective`	value of the log-likelihood at the maximum.
`convergence`	an integer code. 0 indicates successful convergence, see the documentation of nlminb.
`iterations`	number of iterations, see the documentation of nlminb.
`evaluations`	number of evaluations of the objective and gradient functions, see the documentation of nlminb.
`message`	a character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation and the nlminb documentation.
`NOTE`	an additional message returned if one tries to estimate a 2-component model without leverage.

Note

Empty

Author(s)

Genaro Sucarrat, http://www.sucarrat.net/

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

Nelson, Daniel B. (1991): 'Conditional Heteroskedasticity in Asset Returns: A New Approach', Econometrica 59, pp. 347-370.

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.

Schwarz (1978), 'Estimating the Dimension of a Model', The Annals of Statistics 6, pp. 461-464.

Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

See Also

`tegarchSim`, `coef.tegarch`, `fitted.tegarch`, `logLik.tegarch`, `predict.tegarch`, `print.tegarch`, `residuals.tegarch`, `summary.tegarch`, `vcov.tegarch`

Examples

```
##simulate series with 500 observations:
set.seed(123)
y <- tegarchSim(500, omega=0.01, phi1=0.9, kappa1=0.1, kappastar=0.05,
  df=10, skew=0.8)

##estimate a 1st. order Beta-t-EGARCH model and store the output in mymod:
mymod <- tegarch(y)

#print estimates and standard errors:
print(mymod)

#graph of fitted volatility (conditional standard deviation):
plot(fitted(mymod))

#graph of fitted volatility and more:
plot(fitted(mymod, verbose=TRUE))

#plot forecasts of volatility 1-step ahead up to 20-steps ahead:
plot(predict(mymod, n.ahead=20))

#full variance-covariance matrix:
vcov(mymod)
```

`tegarchLogl` *Auxiliary functions*

Description

tegarchLogl, tegarchLogl2, tegarchRecursion and tegarchRecursion2 are auxiliary functions called by `tegarch`, and which are not intended to be used for the average user. Henceforth they are thusonly scarcely documented, but most should either be self-explanatory (for the non-average user!) or more or less documented in relation with the `tegarch` and `tegarchSim` functions.

Usage

```
##the '2' relates to the 2-component specification:
tegarchLogl(y, pars, lower = -Inf, upper = Inf, lambda.initial = NULL,
  logl.penalty = -1e+100, c.code = TRUE, aux = NULL)
tegarchLogl2(y, pars, lower = -Inf, upper = Inf, lambda.initial = NULL,
  logl.penalty = -1e+101, c.code = TRUE, aux = NULL)
tegarchRecursion(y, omega = 0.1, phi1 = 0.4, kappa1 = 0.2, kappastar = 0.1,
  df = 10, skew = 0.6, lambda.initial = NULL, c.code = TRUE, verbose = FALSE,
  aux = NULL)
tegarchRecursion2(y, omega = 0.1, phi1 = 0.4, phi2 = 0.2, kappa1 = 0.05,
  kappa2 = 0.1, kappastar = 0.02, df = 10, skew = 0.6, lambda.initial = NULL,
  c.code = TRUE, verbose = FALSE, aux = NULL)
```

Arguments

`y`	numeric vector, typically a financial return series
`omega`	numeric
`phi1`	numeric, must be less than 1 in absolute value
`phi2`	numeric, must be less than 1 in absolute value
`kappa1`	numeric
`kappa2`	numeric
`kappastar`	numeric
`df`	numeric, the value of df (degrees of freedom)
`skew`	numeric (positive), the value of skew (skewness parameter)
`verbose`	logical. If FALSE (default) then only lambda is returned. If TRUE then a matrix with y and the fitted values of, amongst other, sigma, the log-scale (lambda), the conditional standard deviation (stdev), u, epsilon and the standardised residuals (residstd) are returned
`pars`	numeric vector, the parameter values
`lower`	numeric vector, the lower bounds used during estimation
`upper`	numeric vector, the upper bounds used during estimation
`lambda.initial`	NULL (default) or initial value(s) of the recursion for lambda. If NULL, then the values are chosen automatically
`logl.penalty`	numeric value
`c.code`	logical. TRUE (default) is faster since it makes use of compiled C-code
`aux`	NULL (default) or a list, se `tegarch` code

Details

tegarchLogl and tegarchLogl2 return the value of the log-likelihood for a 1-component and 2-component model, respectively.

Value

`tegarchLogl`: The log-likelihood value (i.e. a numeric) of a 1-component specification

`tegarchLogl2`: The log-likelihood value (i.e. a numeric) of a 2-component specification

`tegarchRecursion`: A numeric vector containing the lambda values if verbose=FALSE (default). If verbose=TRUE then a matrix then a matrix with y and the fitted values of sigma, the log-scale (lambda), the conditional standard deviation (stdev), u, epsilon and the standardised residuals (residstd) are returned

`tegarchRecursion2`: A numeric vector containing the lambda values if verbose=FALSE (default). If verbose=TRUE, then a matrix then a matrix with y and the fitted values of sigma, the log-scale (lambda), the conditional standard deviation (stdev), u, epsilon and the standardised residuals (residstd) are returned

Author(s)

Genaro Sucarrat, http://www.sucarrat.net/

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.

Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

See Also

`tegarch`, `tegarchSim`, `fitted.tegarch`

`tegarchSim` *Simulate from a first order Beta-Skew-t-EGARCH model*

Description

Simulate the y series (typically interpreted as a financial return or the error in a regression) from a first order Beta-Skew-t-EGARCH model. Optionally, the conditional scale (sigma), log-scale (lambda), conditional standard deviation (stdev), dynamic components (lambdadagger in the 1-component specification, lambda1dagger and lambda2dagger in the 2-component specification), score (u) and centred innovations (epsilon) are also returned.

Usage

```
tegarchSim(n, omega = 0, phi1 = 0.95, phi2 = 0, kappa1 = 0.01, kappa2 = 0,
  kappastar = 0, df = 10, skew = 1, lambda.initial = NULL, verbose = FALSE)
```

Arguments

`n`	integer, length of y (i.e. no of observations)
`omega`	numeric, the value of omega
`phi1`	numeric, the value of phi1
`phi2`	numeric, the value of phi2
`kappa1`	numeric, the value of kappa1
`kappa2`	numeric, the value of kappa2
`kappastar`	numeric, the value of kappastar
`df`	numeric, the value of df (degrees of freedom)
`skew`	numeric, the value of skew (skewness parameter
`lambda.initial`	NULL (default) or initial value(s) of the recursion for lambda or log-volatility. If NULL then the values are chosen automatically
`verbose`	logical, TRUE or FALSE (default). If TRUE then a matrix with n rows containing y, sigma, lambda, lambdadagger, u and epsilon is returned. If FALSE then only y is returned

Details

Empty

Value

A `zoo` vector of length n or a `zoo` matrix with n rows, depending on the value of verbose.

Author(s)

Genaro Sucarrat, `http://www.sucarrat.net/`

References

Fernandez and Steel (1998), 'On Bayesian Modeling of Fat Tails and Skewness', Journal of the American Statistical Association 93, pp. 359-371.

Harvey and Sucarrat (2014), 'EGARCH models with fat tails, skewness and leverage'. Computational Statistics and Data Analysis 76, pp. 320-338.

Sucarrat (2013), 'betategarch: Simulation, Estimation and Forecasting of First-Order Beta-Skew-t-EGARCH models'. The R Journal (Volume 5/2), pp. 137-147.

See Also

tegarch, zoo

Examples

```
##1-component specification: simulate series with 500 observations:
set.seed(123)
y <- tegarchSim(500, omega=0.01, phi1=0.9, kappa1=0.1, kappastar=0.05,
  df=10, skew=0.8)

##simulate the same series, but with more output (volatility, log-volatility or
##lambda, lambdadagger, u and epsilon)
set.seed(123)
y <- tegarchSim(500, omega=0.01, phi1=0.9, kappa1=0.1, kappastar=0.05, df=10, skew=0.8,
  verbose=TRUE)

##plot the simulated values:
plot(y)

##2-component specification: simulate series with 500 observations:
set.seed(123)
y <- tegarchSim(500, omega=0.01, phi1=0.95, phi2=0.9, kappa1=0.01, kappa2=0.05,
  kappastar=0.03, df=10, skew=0.8)
```

Index

∗ **Statistical Models**
  coef.tegarch, 3
  dST, 6
  predict.tegarch, 8
  tegarch, 10
  tegarchLogl, 12
  tegarchSim, 14
∗ **datasets**
  nasdaq, 7

betategarch (betategarch-package), 2
betategarch-package, 2

coef, *5*
coef.tegarch, *3*, 3, *12*

dST, 6

fitted, *5*
fitted.tegarch, *3*, *12*, *14*
fitted.tegarch (coef.tegarch), 3

logLik, *5*
logLik.tegarch, *3*, *12*
logLik.tegarch (coef.tegarch), 3

nasdaq, 7
nlminb, *10*

predict, *5*, *9*
predict.tegarch, *3*, *5*, 8, *12*
print, *5*
print.tegarch, *3*, *12*
print.tegarch (coef.tegarch), 3

residuals.tegarch, *3*, *12*
residuals.tegarch (coef.tegarch), 3
rST (dST), 6

STkurtosis (dST), 6
STmean (dST), 6
STskewness (dST), 6
STvar (dST), 6
summary, *5*
summary.tegarch, *3*, *12*
summary.tegarch (coef.tegarch), 3

tegarch, *2*, *5*, *8, 9*, 10, *12–14*, *16*
tegarchLogl, 12
tegarchLogl2 (tegarchLogl), 12
tegarchRecursion (tegarchLogl), 12
tegarchRecursion2 (tegarchLogl), 12
tegarchSim, *2, 3*, *7*, *12*, *14*, 14

vcov, *5*
vcov.tegarch, *3*, *12*
vcov.tegarch (coef.tegarch), 3

zoo, *3*, *15, 16*