Package ‘garchx’

July 15, 2021

Type Package
Title Flexible and Robust GARCH-X Modelling
Version 1.3
Date 2021-07-15
Author Genaro Sucarrat [aut, cre]
Maintainer Genaro Sucarrat <gsucarrat@gmail.com>
Description Flexible and robust estimation and inference of generalised autoregressive conditional heteroscedasticity (GARCH) models with covariates ('X') based on the results by Francq and Thieu (2018) <doi:10.1017/S0266466617000512>. Coefficients can straightforwardly be set to zero by omission, and quasi maximum likelihood methods ensure estimates are generally consistent and inference valid, even when the standardised innovations are non-normal and/or dependent over time.
License GPL (>= 2)
Depends R (>= 3.4.0), zoo
BugReports https://github.com/gsucarrat/garchx/issues
URL http://www.sucarrat.net/
NeedsCompilation yes
Repository CRAN
Date/Publication 2021-07-15 14:10:02 UTC

R topics documented:

garchx-package ............................................... 2
goef.garchx .................................................. 3
garchx ......................................................... 5
garchxAvar ................................................... 8
garchxObjective ............................................. 9
garchxSim ..................................................... 10
gdiff ......................................................... 11
glag ........................................................... 12
rmnorm ....................................................... 13
ttest0 ......................................................... 14
Flexible and robust estimation and inference of GARCH(p,q,r)-X models, where p is the ARCH order, q is the GARCH order, r is the asymmetry or leverage order, and 'X' indicates that covariates can be included. Suitable subsets of the coefficients can be restricted to zero by omission, and Quasi Maximum Likelihood (QML) methods ensure estimates are generally consistent, even when the standardised innovations are non-normal and/or dependent.

Details

Package: garchx
Type: Package
Version: 1.3
Date: 2021-07-15
License: GPL-2

Author(s)

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

Maintainer: Genaro Sucarrat

See Also

garchxSim, coef, fitted, logLik, print, residuals, vcov

Examples

##simulate from a garch(1,1):
set.seed(123)
y <- garchxSim(1000)

##estimate garch(1,1) model:
mymod <- garchx(y)
mymod
Description

Extraction functions for objects of class ‘garchx’

Usage

```r
## S3 method for class 'garchx'
coef(object, ...)
## S3 method for class 'garchx'
fitted(object, as.zoo = TRUE, ...)
## S3 method for class 'garchx'
logLik(object, ...)
## S3 method for class 'garchx'
nobs(object, ...)
## S3 method for class 'garchx'
predict(object, n.ahead = 10, newxreg = NULL,
    newindex = NULL, n.sim = NULL, verbose = FALSE, ...)
## S3 method for class 'garchx'
print(x, ...)
## S3 method for class 'garchx'
quantile(x, probs = 0.025, names = TRUE, type = 7, as.zoo = TRUE, ...)
## S3 method for class 'garchx'
residuals(object, as.zoo = TRUE, ...)
## S3 method for class 'garchx'
toLatex(object, digits = 4, ...)
## S3 method for class 'garchx'
vcov(object, vcov.type = NULL, ...)
```

Arguments

- `object`: an object of class ‘garchx’
- `x`: an object of class ‘garchx’
- `as.zoo`: logical. If TRUE, then the returned result is of class `zoo`
- `n.ahead`: integer that determines how many steps ahead predictions should be generated
- `newxreg`: vector or matrix with the out-of-sample regressor values
- `newindex`: zoo-index for the out-of-sample predictions. If NULL (default), then 1:n.ahead is used
- `n.sim`: NULL or an integer, the number of simulations
- `verbose`: logical. If TRUE, then the simulations - in addition to the predictions - are returned
- `probs`: vector of probabilities
names logical, whether to return names or not
type integer that determines the algorithm used to compute the quantile, see quantile
digits integer, the number of digits in the printed LaTeX code
vcov.type NULL or a character that is (partially) matched to "ordinary" or "robust". The robust coefficient-covariance is that of Francq and Thieu (2018).
... additional arguments

Value
coef: numeric vector containing parameter estimates
fitted: fitted conditional variance
logLik: log-likelihood (normal density)
nobs: the number of observations used in the estimation
predict: a vector with the predictions (verbose=FALSE), or a matrix with both the predictions and the simulations (verbose=TRUE)
print: print of the estimation results
quantile: the fitted quantiles, i.e. the conditional standard deviation times the empirical quantile of the standardised innovations
residuals: standardised residuals
vcov: coefficient variance-covariance matrix

Author(s)
Genaro Sucarrat, http://www.sucarrat.net/

References

See Also
garchx, garchxSim, zoo

Examples

##simulate from a garch(1,1):
set.seed(123)
y <- garchxSim(1000)

##estimate garch(1,1) model:
mymod <- garchx(y)

##print estimation results:
print(mymod)

##extract coefficients:
Estimate a GARCH-X model

Quasi Maximum Likelihood (ML) estimation of a GARCH(p,q,r)-X model, where p is the ARCH order, q is the GARCH order, r is the asymmetry (or leverage) order and 'X' indicates that covariates can be included. Note that the underlying estimation theory assumes the covariates are stochastic. The estimation procedure will, in general, provide consistent estimates when the standardised innovations are not normal or independent (or both), see Francq and Thieu (2018).

Usage

garchx(y, order = c(1,1), arch = NULL, garch = NULL, asym = NULL, xreg = NULL, vcov.type = c("ordinary", "robust"), initial.values = NULL, backcast.values = NULL, lower = 0, upper = +Inf, control = list(), hessian.control = list(), solve.tol = .Machine$double.eps, estimate = TRUE, c.code = TRUE, penalty.value = NULL, sigma2.min = .Machine$double.eps, objective.fun = 1, turbo = FALSE)

Arguments

y numeric vector, time-series or zoo object. Missing values in the beginning and at the end of the series is allowed, as they are removed with the na.trim command

order integer vector of length 1, 2 or 3, for example c(1,1,1). The first entry controls the GARCH order, the second the ARCH order and the third the ASYM (asymmetry/leverage) order

arch NULL or numeric vector containing the ARCH-terms to include. Note: If not NULL, then the value of the ARCH argument overrides the value of the first entry in the order argument
garchx

**garch**
NULL or numeric vector containing the GARCH-terms to include. Note: If not NULL, then the value of the GARCH argument overrides the value of the second entry in the order argument.

**asym**
NULL or numeric vector containing the ASYM-terms (asymmetry/leverage terms) to include. Note: If not NULL, then the value of the ASYM argument overrides the value of the third entry in the order argument.

**xreg**
numeric vector, time-series or zoo object. Missing values in the beginning and at the end of the series is allowed, as they are removed with the na.trim command.

**vcov.type**
character, either "ordinary" or "robust", see vcov.garchx.

**initial.values**
NULL or a numeric vector with the initial parameter values passed on to the optimisation routine, nlminb. If NULL, the default, then the values are chosen automatically.

**backcast.values**
NULL or a non-negative numeric. The backcast value is used to initiate the forward recursion of the conditional variance. If NULL (default), then the value is chosen automatically (currently the average of y squared is used). If backcast.values is a non-negative numeric, then the initial recursion values are all set to this value.

**lower**
numeric vector, either of length 1 or the number of parameters to be estimated, see nlminb.

**upper**
numeric vector, either of length 1 or the number of parameters to be estimated, see nlminb.

**control**
a list passed on to the control argument of nlminb.

**hessian.control**
a list passed on to the control argument of optimHess.

**solve.tol**
numeric value passed on to the tol argument of solve, which is called whenever the coefficient variance-covariance matrix is computed. The value controls the tolerance for detecting linear dependence between columns when inverting a matrix.

**estimate**
logical, if TRUE then estimation is carried out. If FALSE, then the initial.values are used.

**c.code**
logical, if TRUE then compiled C code is used in the forward recursion.

**penalty.value**
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. The penalty.value is the value returned by the objective function garchxObjective in the presence of NA or +/-Inf values.

**sigma2.min**
numeric with default .Machine$double.eps. To avoid taking taking the log of a very small value when computing the log-likelihood, sigma2.min is used as the lower bound of the fitted conditional variances, see the code of garchxObjective.

**objective.fun**
numeric, either 1 or 0.

**turbo**
logical. If FALSE (default), then the coefficient variance-covariance is computed during estimation, and the fitted values and residuals are attached to the returned object. If TRUE, then these operations are skipped, and hence estimation is faster. Note, however, that if turbo is set to TRUE, then the coefficient-covariance, fitted values and residuals can still be extracted subsequent to estimation with vcov.garchx, fitted.garchx and residuals.garchx, respectively.
Value

A list of class 'garchx'

Author(s)

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

References


See Also

  `garchxSim`, `nlminb`, `optimHess`, `coef.garchx`

Examples

```r
# simulate from a garch(1,1):
set.seed(123)
y <- garchxSim(1000)

# estimate garch(1,1) model:
mymod <- garchx(y)

# print estimation results:
print(mymod)

# extract coefficients:
coef(mymod)

# extract and store conditional variances:
sigma2hat <- fitted(mymod)

# extract log-likelihood:
logLik(mymod)

# extract and store standardised residuals:
etahat <- residuals(mymod)

# extract variance-covariance matrix:
vcov(mymod)

# generate predictions:
predict(mymod)
```
garchxAvar  
Asymptotic Coefficient Covariance

Description

Compute the asymptotic coefficient-covariance of a GARCH(p,q,r)-X model by simulation. Note that the principles of how to use the arch, garch, asym and xreg arguments are the same as those of garchx

Usage

garchxAvar(pars, arch = NULL, garch = NULL, asym = NULL, xreg = NULL, vcov.type = c("ordinary", "robust"), innovations = NULL, Eeta4 = NULL, n = 1e+06, objective.fun = 1, seed = NULL)

Arguments

pars vector of parameters of length 1 or more. The first component contains the coefficient-value of the intercept, the next component(s) the ARCH-coefficient(s), and so on.
arch NULL or integer vector with the lags of the ARCH-terms to include. Works in the same way as the arch argument in the garchx function
garch NULL or integer vector with the lags of the GARCH-terms. Works in the same way as the garch argument in the garchx function
asym NULL or integer vector with the lags of the asymmetry terms to include. Works in the same way as the asym argument in the garchx function
xreg NULL, or a vector or matrix with the covariates of the model. Works in the same way as the xreg argument in the garchx function
vcov.type character that determines the type of coefficient-covariance
innovations NULL or a vector with the standardised innovations to use. If NULL, then the innovations are standard normal
Eeta4 numeric, the fourth moment of the innovations. If NULL, then the value is estimated internally. Note: The value of Eeta4 is only used if vcov.type = "ordinary", otherwise it is ignored
n integer, the number of observations to use in the simulations
objective.fun integer equal to 1 or 0 that determines the type of objective function to use, see the code of garchxObjective
seed NULL or an integer that sets the seed (the value is passed on to set.seed. Useful for reproducibility

Value

A matrix
**garchxObjective**

**Author(s)**
Genaro Sucarrat, http://www.sucarrat.net/

**References**

**See Also**
garchx, garchxSim, vcov.garchx

**Examples**

```r
# asymptotic coefficient-covariance of a garch(1,1)
# note: the estimate is rough, since n is small
intercept <- 0.2
alpha <- 0.1
beta <- 0.8
pars <- c(intercept, alpha, beta)
seed <- 123 # for reproducibility
garchxAvar(pars, arch=1, garch=1, n=10000, seed=seed)
```

**Description**
Auxiliary functions used in estimation. Not intended for the average user

**Usage**

```r
garchxObjective(pars, aux)
garchxRecursion(pars, aux)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pars</td>
<td>numeric vector of parameters</td>
</tr>
<tr>
<td>aux</td>
<td>list created by garchx</td>
</tr>
</tbody>
</table>

**Value**

- garchxObjective: value of the objective function
- garchxRecursion: fitted conditional variance
garchxSim

Simulate from a GARCH-X model

Description

Simulate from a GARCH(p,q,r)-X model. Optionally, if verbose=TRUE, the conditional variance and innovations are also returned.

Usage

garchxSim(n, intercept = 0.2, arch = 0.1, garch = 0.8, asym = NULL, xreg = NULL, innovations = NULL, backcast.values = list(), verbose = FALSE, as.zoo = TRUE)

Arguments

n integer
intercept numeric
arch NULL or numeric vector with the values of the ARCH-coefficients
garch NULL or numeric vector with the values of the GARCH-coefficients
asym NULL or numeric vector with the values of the asymmetry-coefficients
xreg NULL or numeric vector with the values of the X-term
innovations NULL or numeric vector with the innovations. If NULL, then standard normal innovations are generated with rnorm
backcast.values list with backcast values
verbose logical
as.zoo logical. If TRUE (default), then the returned object is of class zoo

Value

a numeric vector or matrix with the simulated values.

Author(s)

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

*Difference a vector or a matrix, with special treatment of zoo objects*

**Description**

Similar to the `diff` function from the base package, but `gdiff` enables padding (e.g. NAs or 0s) of the lost entries. Contrary to the `diff` function in the base package, however, the default in `gdiff` is to pad (with NAs). The `gdiff` function is particularly suited for `zoo` objects, since their indexing is retained.

**Usage**

```r
gdiff(x, lag = 1, pad = TRUE, pad.value = NA)
```

**Arguments**

- `x` a numeric vector or matrix
- `lag` integer equal to the difference-length (the default is 1)
- `pad` logical. If `TRUE` (default), then the lost entries are padded with `pad.value`. If `FALSE`, then no padding is undertaken
- `pad.value` numeric, the pad-value

**Value**

A vector or matrix with the differenced values

**Note**

Empty

**Author(s)**

Genaro Sucarrat, [http://www.sucarrat.net/](http://www.sucarrat.net/)
See Also

diff, glag, lag

Examples

```r
# 1st difference of a series:
x <- rnorm(5)
gdiff(x)

# 1st difference with no padding:
gdiff(x, pad = FALSE)

# 1st difference retaining the original zoo-index ordering:
gdiff(as.zoo(x))

# 1st difference of a matrix:
y <- matrix(rnorm(8), 4, 2)
gdiff(y)

# 2nd difference of the same matrix:
gdiff(y, lag = 2)
```

---

glag

Lag a vector or a matrix, with special treatment of zoo objects

Description

Similar to the lag function from the stats package, but glag enables padding (e.g. NAs or 0s) of the lost entries. Contrary to the lag function in the stats package, however, the default in glag is to pad (with NAs). The glag is particularly suited for zoo objects, since their indexing is retained.

Usage

```r
glag(x, k = 1, pad = TRUE, pad.value = NA)
```

Arguments

- `x`: a numeric vector or matrix
- `k`: integer equal to the lag (the default is 1)
- `pad`: logical. If TRUE (default), then the lost entries are padded with pad.value. If FALSE, then no padding is undertaken
- `pad.value`: the pad-value

Value

A vector or matrix with the lagged values
Note

Empty

Author(s)

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

See Also

lag, gdiff, diff

Examples

## lag series with NA for the missing entries:
x <- rnorm(5)
glag(x)

## lag series with no padding:
x <- rnorm(5)
glag(x, pad=FALSE)

## lag series and retain the original zoo-index ordering:
x <- as.zoo(rnorm(5))
glag(x)

## lag two periods:
glag(x, k=2)

rmnorm

Random number generation from the multivariate normal distribution

Description

This function is a speed-optimised version of the rmnorm function from the mnormt package of Adelchi Azzalini (2013).

Usage

rmnorm(n, mean = NULL, vcov = 1)

Arguments

n integer, the number of observations to generate
mean numeric vector, i.e. the mean values
vcov numeric matrix, i.e. the variance-covariance matrix
Value

A matrix of n rows

Author(s)

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

References


Examples

```r
## generate from univariate standardised normal:
z1 <- rmnorm(100)

## generate from bivariate, independent standardised normal:
z2 <- rmnorm(100, vcov=diag(c(1,1)))

## generate from bivariate, dependent standardised normal:
z3 <- rmnorm(100, vcov=cbind(c(1,0.3),c(0.3,1)))
```

---

**ttest0**

* T-tests and Wald-tests under nullity

Description

The permissible parameter-space of GARCH-models is bounded from below by 0. This means non-standard inference is required when one or more parameters are 0 under the null hypothesis. The functions `ttest0` and `waldtest0` perform t-tests and Wald-tests when one or more parameters is 0. In the latter test, the Wald-test, the critical values are obtained by simulation, see Francq and Thieu (2018).

Usage

```r
ttest0(x, k = NULL)
waldtest0(x, r = 0, R = NULL, level = c(0.1,0.05,0.01),
          vcov.type = NULL, quantile.type = 7, n = 20000)
```

Arguments

- **x**: an object of class 'garchx'
- **k**: `NULL` (default) or a vector of integers with the coefficients to test. If `NULL`, then all coefficients apart from the intercepts are tested
- **r**: vector with restrictions
The `ttest0` function performs a t-test of coefficient $k$ with 0 as null. Under this null the parameter is on the boundary of the admissible parameter space, and so the distribution is non-standard under the null. The function `ttest0` returns the result(s) of these non-standard t-test(s), see Francq and Thieu (2018). If `k=NULL`, the default, then a test for each coefficient apart from the intercept is undertaken.

The `waldtest0` function performs a Wald-test of the restrictions in $r$, when one or more of its elements are 0, see Francq and Thieu (2018).

**Value**

`ttest0`: a matrix with the t-tests

`waldtest0`: a list with the test-statistic and the critical values

**Author(s)**

Genaro Sucarrat, [http://www.sucarrat.net/](http://www.sucarrat.net/)

**References**


**See Also**

`garchx`, `quantile`, `vcov.garchx`, `rmnorm`

**Examples**

```r
##simulate and estimate a garch(1,1):
set.seed(123)
y <- garchxSim(1000)
mymod <- garchx(y)

##t-tests:
ttest0(mymod)

##wald-test:
waldtest0(mymod)
```
Index

* Econometrics
  coef.garchx, 3
garchx, 5
garchx-package, 2
garchxAvar, 8
garchxObjective, 9
garchxSim, 10
gdiff, 11
glag, 12
rmnorm, 13
ttest0, 14

* Financial Econometrics
  coef.garchx, 3
garchx, 5
garchx-package, 2
garchxAvar, 8
garchxObjective, 9
garchxSim, 10
gdiff, 11
glag, 12
rmnorm, 13
ttest0, 14

* Statistical Models
  coef.garchx, 3
garchx, 5
garchx-package, 2
garchxAvar, 8
garchxObjective, 9
garchxSim, 10
gdiff, 11
glag, 12
rmnorm, 13
ttest0, 14

* Time Series
  coef.garchx, 3
garchx, 5
garchx-package, 2
garchxAvar, 8
garchxObjective, 9
garchxSim, 10
gdiff, 11
glag, 12
rmnorm, 13
ttest0, 14

diff, 11–13
fitted, 2
fitted.garchx, 6, 10
fitted.garchx (coef.garchx), 3
garchx, 4, 5, 8–11, 15
garchx-package, 2
garchxAvar, 8
garchxObjective, 6, 8, 9
garchxRecursion (garchxObjective), 9
garchxSim, 2, 4, 7, 9, 10
gdiff, 11, 13
glag, 12, 12
lag, 12, 13
list, 6, 9, 10
logLik, 2
logLik.garchx (coef.garchx), 3
na.trim, 5, 6
nlminb, 6, 7
nobs.garchx (coef.garchx), 3
optimHess, 6, 7
predict.garchx (coef.garchx), 3
print, 2
print.garchx (coef.garchx), 3
quantile, 4, 15
quantile.garchx (coef.garchx), 3
residuals, 2
residuals.garchx, 6, 10
residuals.garchx (coef.garchx), 3
rmnorm, 13, 13, 15
rnorm, 10

set.seed, 8
solve, 6

toLatex.garchx (coef.garchx), 3
ttest0, 14

vcov, 2
vcov.garchx, 6, 9, 15
vcov.garchx (coef.garchx), 3

waldtest0 (ttest0), 14

zoo, 3–6, 10–12