

# Package ‘garchx’

June 30, 2026

**Type** Package

**Title** Flexible and Robust GARCH-X Modelling

**Version** 1.7

**Date** 2026-06-29

**Author** Genaro Sucarrat [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-8433-837X>>)

**Maintainer** Genaro Sucarrat <[gsucarrat@gmail.com](mailto: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 (2019) <[doi:10.1017/S0266466617000512](https://doi.org/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. See Sucarrat (2021) <[doi:10.32614/RJ-2021-057](https://doi.org/10.32614/RJ-2021-057)> for an overview of the package.

**License** GPL (>= 2)

**Depends** R (>= 3.4.0), methods, zoo

**Suggests** tvgarch, lgarch

**BugReports** <https://github.com/gsucarrat/garchx/issues>

**URL** <https://www.sucarrat.net/>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2026-06-30 07:00:02 UTC

## Contents

garchx-package . . . . .	2
coef.garchx . . . . .	3
garchx . . . . .	5
garchxAvar . . . . .	8
garchxObjective . . . . .	10

garchxSim . . . . .	11
gdiff . . . . .	12
glag . . . . .	13
refit . . . . .	14
rmnorm . . . . .	16
ttest0 . . . . .	17

<b>Index</b>	<b>19</b>
--------------	-----------

---

garchx-package	<i>Flexible and Robust GARCH-X Modelling</i>
----------------	--

---

## Description

Flexible and robust estimation and inference of GARCH( $q,p,r$ )-X models, where  $q$  is the GARCH order,  $p$  is the ARCH 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: 1.7  
 Date: 2026-06-29  
 License: GPL-2

## Author(s)

Genaro Sucarrat, <https://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
```

coef.garchx

*Extraction functions for 'garchx' objects***Description**

Extraction functions for objects of class 'garchx'

**Usage**

```
## S3 method for class 'garchx'
coef(object, ...)
## S3 method for class 'garchx'
confint(object, parm, level = 0.95, ...)
## 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, bw = NULL,
      kernel.weights = NULL, ...)
```

**Arguments**

object            an object of class 'garchx'

x	an object of class 'garchx'
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered
level	numeric value between 0 and 1 (the confidence level required)
as.zoo	logical. If TRUE, then the returned result is of class <code>zoo</code>
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	<code>zoo</code> -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 <code>quantile</code>
digits	integer, the number of digits in the printed LaTeX code
vcov.type	NULL or character that is (partially) matched to "ordinary", "robust" or "hac". The robust coefficient-covariance ("robust") is that of Francq and Thieu (2019). The Heteroscedasticity and Autocorrelation Consistent ("hac") variance-covariance estimates the variance of the score (the 'meat') based on Theorem 2.2 in De Jong and Davidson (2000). For the details of the kernel weights (Bartlett) and bandwidth, see the code in the <code>vcov.garchx</code> function
bw	NULL or non-negative numeric value, the bandwidth of the HAC coefficient-covariance. If NULL, then it is computed internally by <code>vcov.garchx</code>
kernel.weights	NULL (default) or a non-negative numeric vector with kernel weights for the HAC coefficient-covariance estimator. If NULL, then Bartlett weights are used, see the code of the <code>vcov.garchx</code> function
...	additional arguments

### Value

coef:	numeric vector containing parameter estimates
confint:	A matrix lower and upper confidence limits for each parameter
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, <https://www.sucarrat.net/>

**References**

Christian Francq and Le Quien Thieu (2019): 'QML inference for volatility models with covariates', *Econometric Theory* 35, pp. 37-72, doi:10.1017/S0266466617000512 Robert M. de Jong and James Davidson (2000): 'Consistency of Kernel Estimators of Heteroscedastic and Autocorrelated Covariance Matrices', *Econometrica* 68, pp. 407-423

**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:
coef(mymod)

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

##extract log-likelihood:
logLik(mymod)

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

##extract coefficient variance-covariance matrix:
vcov(mymod)

##generate predictions:
predict(mymod)
```

## Description

Quasi Maximum Likelihood (ML) estimation of a GARCH( $q,p,r$ )-X model, where  $q$  is the GARCH order,  $p$  is the ARCH 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", "hac"), bw = NULL,
       kernel.weights = NULL, 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

<code>y</code>	numeric vector, time-series or <a href="#">zoo</a> object. Missing values in the beginning and at the end of the series is allowed, as they are removed with the <a href="#">na.trim</a> command
<code>order</code>	integer vector of length 1, 2 or 3, for example <code>c(1,1,1)</code> . The first entry controls the GARCH order, the second the ARCH order and the third the ASYM (asymmetry/leverage) order
<code>arch</code>	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
<code>garch</code>	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
<code>asym</code>	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
<code>xreg</code>	numeric vector, time-series or <a href="#">zoo</a> object. Missing values in the beginning and at the end of the series is allowed, as they are removed with the <a href="#">na.trim</a> command
<code>vcov.type</code>	character, either "ordinary", "robust" or "hac". The robust coefficient-covariance ("robust") is that of Francq and Thieu (2019). The Heteroscedasticity and Autocorrelation Consistent ("hac") variance-covariance estimates the variance of the score (the 'meat') based on Theorem 2.2 in De Jong and Davidson (2000). For the details of the kernel weights (Bartlett) and bandwidth, see the code in the <a href="#">vcov.garchx</a> function
<code>bw</code>	NULL (default) or a non-negative numeric value, the bandwidth of the HAC coefficient-covariance. If NULL, then it is computed internally by <a href="#">vcov.garchx</a>
<code>kernel.weights</code>	NULL (default) or a non-negative numeric vector with kernel weights for the HAC coefficient-covariance estimator. If NULL, then Bartlett weights are used, see the code of <a href="#">vcov.garchx</a>

<code>initial.values</code>	NULL or a numeric vector with the initial parameter values passed on to the optimisation routine, <code>nlminb</code> . If NULL, the default, then the values are chosen automatically
<code>backcast.values</code>	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 <code>backcast.values</code> is a non-negative numeric, then the initial recursion values are all set to this value
<code>lower</code>	numeric vector, either of length 1 or the number of parameters to be estimated, see <code>nlminb</code>
<code>upper</code>	numeric vector, either of length 1 or the number of parameters to be estimated, see <code>nlminb</code>
<code>control</code>	a <code>list</code> passed on to the <code>control</code> argument of <code>nlminb</code>
<code>hessian.control</code>	a <code>list</code> passed on to the <code>control</code> argument of <code>optimHess</code>
<code>solve.tol</code>	numeric value passed on to the <code>tol</code> argument of <code>solve</code> , which is called whenever the coefficient variance-coariance matrix is computed. The value controls the tolerance for detecting linear dependence between columns when inverting a matrix
<code>estimate</code>	logical, if TRUE then estimation is carried out. If FALSE, then the <code>initial.values</code> are used
<code>c.code</code>	logical, if TRUE then compiled C code is used in the forward recursion
<code>penalty.value</code>	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 $\pm\text{Inf}$ values. The <code>penalty.value</code> is the value returned by the objective function <code>garchxObjective</code> in the presence of NA or $\pm\text{Inf}$ values
<code>sigma2.min</code>	numeric with default <code>.Machine\$double.eps</code> . To avoid taking the log of a very small value when computing the log-likelihood, <code>sigma2.min</code> is used as the lower bound of the fitted conditional variances, see the code of <code>garchxObjective</code>
<code>objective.fun</code>	numeric, either 1 or 0
<code>turbo</code>	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 <code>turbo</code> is set to TRUE, then the coefficient-covariance, fitted values and residuals can still be extracted subsequent to estimation with <code>vcov.garchx</code> , <code>fitted.garchx</code> and <code>residuals.garchx</code> , respectively

**Value**

A list of class 'garchx'

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

## References

Christian Francq and Le Quien Thieu (2019): 'QML inference for volatility models with covariates', *Econometric Theory* 35, pp. 37-72, doi:[10.1017/S0266466617000512](https://doi.org/10.1017/S0266466617000512) Christian Francq and Jean-Michel Zakoian (2019): 'GARCH Models', 2nd Edition, Wiley Robert M. de Jong and James Davidson (2000): 'Consistency of Kernel Estimators of Heteroscedastic and Autocorrelated Covariance Matrices', *Econometrica* 68, pp. 407-423

## See Also

[garchxSim](#), [nlminb](#), [optimHess](#), [coef.garchx](#)

## 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:
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(q,p,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", "hac"),
           innovations = NULL, Eeta4 = NULL, n = 1e+06, objective.fun = 1,
           seed = NULL)
```

**Arguments**

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

**Value**

A matrix

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

**References**

Christian Francq and Le Quien Thieu (2018): 'QML inference for volatility models with covariates', *Econometric Theory*, doi:10.1017/S0266466617000512  
 Christian Francq and Jean-Michel Zakoian (2019): 'GARCH Models', 2nd Edition, Wiley

**See Also**

[garchx](#), [garchxSim](#), [vcov.garchx](#)

**Examples**

```
##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)
```

---

garchxObjective

*Auxiliary functions*

---

**Description**

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

**Usage**

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

**Arguments**

`pars` numeric vector of parameters  
`aux` [list](#) created by [garchx](#)

**Value**

`garchxObjective`:  
value of the objective function  
`garchxRecursion`:  
fitted conditional variance

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

**See Also**

[garchx](#), [fitted.garchx](#), [residuals.garchx](#)

---

garchxSim	<i>Simulate from a GARCH-X model</i>
-----------	--------------------------------------

---

**Description**

Simulate from a GARCH(q,p,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,
  c.code = 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 <code>rnorm</code>
backcast.values	<code>list</code> with backcast values
verbose	logical
as.zoo	logical. If TRUE (default), then the returned object is of class <code>zoo</code>
c.code	logical. If TRUE (default), then compiled C code is used for the recursion (faster). Otherwise R code is used (slower)

**Value**

a numeric vector or matrix with the simulated values.

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

**See Also**

[garchx](#), [zoo](#)

## Examples

```
##simulate from a garch(1,1):
y <- garchxSim(1000)

##simulate from a garch(1,1) with asymmetry/leverage:
yy <- garchxSim(1000, asym=0.1)

##simulate from a garch(1,1) w/user-provided backcast values:
yyy <- garchxSim(1000, backcast.values=list(z2=1, sigma2=0.5))
```

---

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

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

## Arguments

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

## Value

A vector or matrix with the differenced values

## Note

Empty

## Author(s)

Genaro Sucarrat, <https://www.sucarrat.net/>

## See Also

`diff`, `glag`, `lag`

**Examples**

```
##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**

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

**Arguments**

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

**Value**

A vector or matrix with the lagged values

**Note**

Empty

**Author(s)**

Genaro Sucarrat, <https://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)
```

---

refit

*Refit a model to new data*

---

**Description**

Refit a model to new data, typically for cross-validation purposes. Re-estimation (reestimate) is optional (the default is FALSE).

**Usage**

```
##generic:
refit(object, ...)

##S3 method for 'garchx' objects:
## S3 method for class 'garchx'
refit(object, newy = NULL, newxreg = NULL,
       backcast.value = NULL, reestimate = FALSE, ...)
```

**Arguments**

object	an object of class <a href="#">garchx</a>
newy	vector, the new 'y' data, see <a href="#">garchx</a>
newxreg	the new 'xreg' data, if any, see <a href="#">garchx</a>
backcast.value	NULL or a non-negative numeric, see <a href="#">garchx</a>
reestimate	logical. If FALSE (default), then the estimates from object are used on the new data. If TRUE, then the model is re-estimated using the new data
...	further arguments passed to or from other methods

**Details**

`refit.garchx` is a convenience function to facilitate cross-validation and related analyses.

**Value**

A list of class 'garchx'

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

**See Also**

[garchx](#)

**Examples**

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

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

##new data (e.g. 'out-of-sample' or 'test' data):
yy <- garchxSim(100)

##apply the estimates of 'mymod' on yy data:
refit(mymod, newy=yy)
```

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**

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

**Value**

A matrix of `n` rows

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

**References**

Adelchi Azzalini (2013): 'mnormt: The multivariate normal and t distributions', R package version 1.4-7, <https://CRAN.R-project.org/package=mnormt>

**Examples**

```
##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, a frequent situation in empirical applications. 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

```
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

<code>x</code>	an object of class 'garchx'
<code>k</code>	NULL (default) or a vector of integers with the coefficients to test. If NULL, then all coefficients apart from the intercepts are tested
<code>r</code>	vector with restrictions
<code>R</code>	NULL (default) or a full-rank matrix. If NULL, then R is specified such that a test of all coefficients - apart from the intercept - is equal to the restriction <code>r</code> . If <code>length(r)==1</code> , then it is recycled so that its dimension match that of R
<code>level</code>	vector of significance levels whose critical values should be computed
<code>vcov.type</code>	NULL or a character that determines the type of coefficient-covariance to use, see <a href="#">vcov.garchx</a>
<code>quantile.type</code>	integer, the algorithm used to compute the quantile, see <a href="#">quantile</a>
<code>n</code>	integer, the number of simulations used to estimate the critical values

### Details

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

<code>ttest0</code> :	a matrix with the t-tests
<code>waldtest0</code> :	a list with the test-statistic and the critical values

**Author(s)**

Genaro Sucarrat, <https://www.sucarrat.net/>

**References**

Christian Francq and Le Quien Thieu (2019): 'QML inference for volatility models with covariates', *Econometric Theory* 35, pp. 37-72, doi:10.1017/S0266466617000512

**See Also**

[garchx](#), [quantile](#), [vcov.garchx](#), [rmnorm](#)

**Examples**

```
##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, 10
- garchxSim, 11
- gdiff, 12
- glag, 13
- refit, 14
- rmnorm, 16
- ttest0, 17

## \* Financial Econometrics

- coef.garchx, 3
- garchx, 5
- garchx-package, 2
- garchxAvar, 8
- garchxObjective, 10
- garchxSim, 11
- gdiff, 12
- glag, 13
- refit, 14
- rmnorm, 16
- ttest0, 17

## \* Statistical Models

- coef.garchx, 3
- garchx, 5
- garchx-package, 2
- garchxAvar, 8
- garchxObjective, 10
- garchxSim, 11
- gdiff, 12
- glag, 13
- refit, 14
- rmnorm, 16
- ttest0, 17

## \* Time Series

- coef.garchx, 3
- garchx, 5

- garchx-package, 2
- garchxAvar, 8
- garchxObjective, 10
- garchxSim, 11
- gdiff, 12
- glag, 13
- refit, 14
- rmnorm, 16
- ttest0, 17

- coef, 2
- coef.garchx, 3, 8
- confint.garchx (coef.garchx), 3

- diff, 12, 14

- fitted, 2
- fitted.garchx, 7, 10
- fitted.garchx (coef.garchx), 3

- garchx, 5, 5, 8–11, 15, 18
- garchx-package, 2
- garchxAvar, 8
- garchxObjective, 7, 9, 10
- garchxRecursion (garchxObjective), 10
- garchxSim, 2, 5, 8, 10, 11
- gdiff, 12, 14
- glag, 12, 13

- lag, 12–14
- list, 7, 10, 11
- logLik, 2
- logLik.garchx (coef.garchx), 3

- na.trim, 6
- nlminb, 7, 8
- nobs.garchx (coef.garchx), 3

- optimHess, 7, 8

- predict.garchx (coef.garchx), 3

print, [2](#)  
print.garchx (coef.garchx), [3](#)

quantile, [4](#), [17](#), [18](#)  
quantile.garchx (coef.garchx), [3](#)

refit, [14](#)  
residuals, [2](#)  
residuals.garchx, [7](#), [10](#)  
residuals.garchx (coef.garchx), [3](#)  
rmnorm, [16](#), [16](#), [18](#)  
rnorm, [11](#)

set.seed, [9](#)  
solve, [7](#)

toLatex.garchx (coef.garchx), [3](#)  
ttest0, [17](#)

vcov, [2](#)  
vcov.garchx, [4](#), [6](#), [7](#), [10](#), [17](#), [18](#)  
vcov.garchx (coef.garchx), [3](#)

waldtest0 (ttest0), [17](#)

zoo, [4–6](#), [11–13](#)