Package 'tsgarch'

October 12, 2024

Type Package

Title Univariate GARCH Models

Version 1.0.3

Maintainer Alexios Galanos <alexios@4dscape.com>

Depends R (>= 3.5.0), methods, tsmethods (>= 1.0.2)

LinkingTo Rcpp (>= 0.10.6), TMB(>= 1.7.20), RcppEigen

Imports TMB (>= 1.7.20), Rcpp, nloptr, Rdpack, numDeriv, xts, zoo, future.apply, future, progressr, flextable, stats, utils, data.table, tsdistributions, lubridate, sandwich

Description Multiple flavors of the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model with a large choice of conditional distributions. Methods for specification, estimation, prediction, filtering, simulation, statistical testing and more. Represents a partial re-write and re-think of 'rugarch', making use of automatic differentiation for estimation.

License GPL-2

Encoding UTF-8

LazyData true

BugReports https://github.com/tsmodels/tsgarch/issues

RdMacros Rdpack

RoxygenNote 7.3.2

ByteCompile true

URL https://github.com/tsmodels/tsgarch

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Alexios Galanos [aut, cre, cph] (<https://orcid.org/0009-0000-9308-0457>)

Repository CRAN

Date/Publication 2024-10-12 00:50:02 UTC

Contents

`+.tsgarch.spec` *Combine univariate GARCH specifications into a multi-specification object*

Description

Combine univariate GARCH specifications into a multi-specification object

Usage

```
## S3 method for class 'tsgarch.spec'
x + y
```

Arguments

x	an object of class "tsgarch.spec"
y	an object of class "tsgarch.spec"

Details

A simple method for combining multiple specifications into an object which can then be estimated using parallel resources. Note that the returned object is effectively a validated list of specification objects with no names. Names can be assigned post-construction (see example).

Value

an object of class "tsgarch.multispec"

Examples

```
library(xts)
x <- xts(rnorm(1000), as.Date(1:1000, origin = "1970-01-01"))
y <- xts(rnorm(1000), as.Date(1:1000, origin = "1970-01-01"))
z <- xts(rnorm(1000), as.Date(1:1000, origin = "1970-01-01"))
mspec <- garch_modelspec(x, model = "egarch") +
garch_modelspec(y, model = "cgarch") +
garch_modelspec(z, model = "aparch")
names(mspec) <- c("x", "y", "z")
sapply(mspec, function(x) x$model$model)
```

AIC.tsgarch.estimate *Akaike's An Information Criterion*

Description

Extract the AIC from an estimated model.

Usage

```
## S3 method for class 'tsgarch.estimate'
AIC(object, ..., k = 2)
```

Arguments

object	an object of class "tsgarch.estimate".
...	not currently used.
k	the penalty per parameter to be used; the default k = 2 is the classical AIC.

Value

A numeric value.

as_flextable.benchmark.laurent *Transform an object into flextable*

Description

Transforms a "benchmark.fcp" or "benchmark.laurent" object into a flextable.

Usage

```
## S3 method for class 'benchmark.laurent'
as_flextable(x, digits = 4, ...)

## S3 method for class 'benchmark.fcp'
as_flextable(x, digits = 4, ...)
```

Arguments

x	an object of class "benchmark.fcp" or "benchmark.aparch".
digits	integer, used for number formatting. Optionally, to avoid scientific notation, set 'options(scipen=999)'.
...	additional arguments passed to flextable method.

Value

A flextable object.

`as_flextable.summary.tsgarch.estimate` *Transform a summary object into flextable*

Description

Transforms a "summary.tsgarch" object into a flextable with options on symbolic representation and model equation.

Usage

```
## S3 method for class 'summary.tsgarch.estimate'
as_flextable(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  include.symbols = TRUE,
  include.equation = TRUE,
  include.statistics = TRUE,
  table.caption = paste0(toupper(x$model), " Model Summary"),
  ...
)
```

Arguments

`x`	an object of class "summary.tsgarch".
`digits`	integer, used for number formatting. Optionally, to avoid scientific notation, set 'options(scipen=999)'.
`signif.stars`	logical. If TRUE, 'significance stars' are printed for each coefficient.
`include.symbols`	logical. If TRUE, replaces parameter names with their symbols (if they exist).
`include.equation`	logical. If TRUE, adds a section with the symbolic model equation.
`include.statistics`	logical. If TRUE, adds a section with summary statistics on the model.
`table.caption`	an optional string for the table caption.
`...`	additional arguments passed to flextable method.

Value

A flextable object.

benchmark_fcp *FCP GARCH Benchmark*

Description

The GARCH(1,1) FCP benchmark.

Usage

```
benchmark_fcp(control = nloptr_fast_options())
```

Arguments

control	control arguments for the nloptr solver.

Details

The benchmark of Fiorentini et al. (1996) on the Deutsche Mark British Pound returns is based on a GARCH(1,1) model with a constant in the conditional mean equation, and normally distributed errors.

Value

An object of class "benchmark.fcp" which has a "as_flextable" method for nice printing of the results.

References

Fiorentini G, Calzolari G, Panattoni L (1996). "Analytic derivatives and the computation of GARCH estimates." *Journal of Applied Econometrics*, **11**(4), 399–417.

benchmark_laurent *Laurent APARCH Benchmark*

Description

The APARCH(1,1) benchmark of Laurent (2003).

Usage

```
benchmark_laurent(control = nloptr_fast_options())
```

Arguments

control	control arguments for the nloptr solver.

Details

The benchmark of Laurent (2003) on the Nikkei daily log returns is based on an APARCH(1,1) model with a constant in the conditional mean equation, and normally distributed errors.

Value

An object of class "benchmark.aparch" which has a "as_flextable" method for nice printing of the results.

References

Laurent S (2004). "Analytical derivates of the APARCH model." *Computational Economics*, **24**, 51–57.

`BIC.tsgarch.estimate` *Bayesian Information Criterion*

Description

Extract the BIC from an estimated model.

Usage

```
## S3 method for class 'tsgarch.estimate'
BIC(object, ...)
```

Arguments

`object`	an object of class "tsgarch.estimate".
`...`	not currently used.

Value

A numeric value.

bread.tsgarch.estimate *Bread Method*

Description

Bread Method

Usage

```
## S3 method for class 'tsgarch.estimate'
bread(x, ...)
```

Arguments

x	an object of class "tsgarch.estimate".
...	not currently used.

Value

The analytic hessian of the model.

Author(s)

Alexios Galanos

coef.tsgarch.estimate *Extract Model Coefficients*

Description

Extract the estimated coefficients of a model.

Usage

```
## S3 method for class 'tsgarch.estimate'
coef(object, ...)
```

Arguments

object	an object of class "tsgarch.estimate".
...	not currently used.

Value

A numeric named vector of estimated coefficients.

confint.tsgarch.estimate *Confidence Intervals for Model Parameters*

Description

Confidence Intervals for Model Parameters

Usage

```
## S3 method for class 'tsgarch.estimate'
confint(object, parm, level = 0.95, vcov_type = "H", ...)
```

Arguments

object	an object of class tsgarch.estimate.
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	the confidence level required.
vcov_type	valid choices are "H" for using the analytic hessian for the bread, "OP" for the outer product of gradients, "QMLE" for the Quasi-ML sandwich estimator (Huber-White), and "NW" for the Newey-West adjusted sandwich estimator (a HAC estimator).
...	additional parameters passed to the Newey-West bandwidth function to determine the optimal lags.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

dmbp *Deutschemark/British pound Exchange Rate*

Description

The Bollerslev-Ghysel benchmark dataset. The variables in the data set are the daily percentage nominal returns computed as 100

$$ln(Pt) - ln(Pt - 1)$$

, where Pt is the bilateral Deutschemark/British pound rate constructed from the corresponding U.S. dollar rates, and a dummy variable that takes the value of 1 on Mondays and other days following no trading in the Deutschemark or British pound/ U.S. dollar market during regular European trading hours, and 0 otherwise. The data spans the period from 1984-01-03 through 1991-12-31, but exact dates are not known as this dataset did not provide an index. This dataset is included as it is used for the GARCH benchmark.

Usage

```
dmbp
```

Format

`dmbp`**:**

A data.frame containing 2x1974 observations

rate The exchange rate

monday Dummy indicator (see descriptiom)

Source

Journal of Business & Economic Statistics Data Archive

`estfun.tsgarch.estimate` *Score Method*

Description

Score Method

Usage

```
## S3 method for class 'tsgarch.estimate'
estfun(x, ...)
```

Arguments

`x`	an object of class "tsgarch.estimate".
`...`	not currently used.

Details

The function returns the scores of the negative of the log likelihood at the optimal solution. The scores are from the second pass of the optimizer using scaling and hence represent the scaled scores. These are then rescaled before returning the matrix.

Value

The score matrix

Author(s)

Alexios Galanos

`estimate.tsgarch.spec` *Estimates an GARCH model given a specification object using maximum likelihood and autodiff*

Description

Estimates an GARCH model given a specification object using maximum likelihood and autodiff

Usage

```
## S3 method for class 'tsgarch.spec'
estimate(
  object,
  solver = "nloptr",
  control = NULL,
  stationarity_constraint = 0.999,
  keep_tmb = FALSE,
  ...
)

## S3 method for class 'tsgarch.multispec'
estimate(
  object,
  solver = "nloptr",
  control = NULL,
  stationarity_constraint = 0.999,
  keep_tmb = FALSE,
  ...
)
```

Arguments

`object`	an object of class "tsgarch.spec" or "tsgarch.multispec"
`solver`	only "nloptr" is currently supported (see `nloptr`).
`control`	solver control parameters.
`stationarity_constraint`	the bound on the inequality constraint for ensuring the stationary of the GARCH process (see details).
`keep_tmb`	whether to save and return the TMB object. This is can be used for example when passing different parameters to the model to extract a new output set given the same dataset (used in "tsmarch" package for calculating partitioned standard errors). The downside is that it can increase the size of the returned object by a factor of 8.
`...`	not currently used.

Details

The underlying code is written using the TMB framework which uses automatic differentiation and hence allows the generation of analytic derivatives.

Stationarity is usually based on the condition that the persistence of the model is less than 1. The argument "stationarity_constraint" allows to fine tune this. For example, setting it to a very high value will effectively render this constraint inactive. The default of 0.999 has been found to be a reasonable bound since values close to one may lead to problems.

Since the nloptr solver make use of analytic Jacobians for the inequality constraint, these are either provided in closed form or calculated as part of the automatic differentiation algorithms implemented in the package.

The estimation makes 2 passes to the solver. The first pass uses no parameter scaling, whilst in the second pass the parameters (as well as bounds) are scaled making use of the estimated hessian from the first pass in order to generate a hopefully more robust solution.

For the multiple specification estimation, parallel functionality is available through the `future_lapply` function. The user needs to setup a `plan` in order to make use of this.

Value

An object of class "tsgarch.estimate" or "tsgarch.multi_estimate".

Author(s)

Alexios Galanos

`fitted.tsgarch.estimate`

Extract Model Fitted Values

Description

Extract the fitted values of the estimated model.

Usage

```
## S3 method for class 'tsgarch.estimate'
fitted(object, ...)

## S3 method for class 'tsgarch.multi_estimate'
fitted(object, ...)
```

Arguments

`object`	an object of class "tsgarch.estimate" or "tsgarch.multi_estimate".
`...`	not currently used.

Value

An xts vector of the fitted values for the univariate type objects. In the case of a multi-estimate object, a list of xts vectors is returned if the individual univariate objects have unequal indices, else an xts matrix is returned.

---

garch_modelspec *GARCH Model Specification*

Description

Specifies a GARCH model prior to estimation.

Usage

```
garch_modelspec(
  y,
  model = "garch",
  constant = FALSE,
  order = c(1, 1),
  variance_targeting = FALSE,
  vreg = NULL,
  multiplicative = FALSE,
  init = c("unconditional", "sample", "backcast"),
  backcast_lambda = 0.7,
  sample_n = 10,
  distribution = "norm",
  ...
)
```

Arguments

y	an xts vector.
model	the type of GARCH model. Valid choices are "garch" for vanilla GARCH, "gjr" for asymmetric GARCH, "egarch" for exponential GARCH, "aparch" for asymmetric power ARCH, "csGARCH" for the component GARCH, "igarch" for the integrated GARCH.
constant	whether to estimate a constant (mean) for y,
order	the (p,q) GARCH order.
variance_targeting	whether to use variance targeting rather than estimating the conditional variance intercept.
vreg	an optional xts matrix of regressors in the conditional variance equation.
multiplicative	whether to exponentiate the contribution of the regressors else will be additive. In the case of the "egarch" model, since this is already a multiplicative model, the regressors are additive irrespective of the choice made.

`init`	the method to use to initialize the recursion of the conditional variance.
`backcast_lambda`	the decay power for the exponential smoothing used when initializing the recursion using the backcast method.
`sample_n`	the number of data points to use when initializing the recursion using the sample method.
`distribution`	a valid distribution from the available re-parameterized distributions of the package.
`...`	not used.

Details

The specification object holds the information and data which is then passed to the maximum likelihood estimation routines.

Value

An object of class "tsgarch.spec".

Author(s)

Alexios Galanos

`halflife.tsgarch.estimate` *Half Life*

Description

Calculates and returns the half-life of a model.

Usage

```
## S3 method for class 'tsgarch.estimate'
halflife(object, ...)
```

Arguments

`object`	an object.
`...`	not currently used.

Details

The half life is defined as the period it takes a series to reach half its long-term average values. For a GARCH model this is defined as $log(0.5)/log(P)$ where P is the persistence.

Value

a numeric value representing the half life in periods based on the frequency of the underlying data.

logLik.tsgarch.estimate *Extract Log-Likelihood*

Description

Extract the log likelihood of the model at the estimated optimal parameter values.

Usage

```
## S3 method for class 'tsgarch.estimate'
logLik(object, ...)
```

Arguments

object	an object of class "tsgarch.estimate".
...	not currently used.

Value

An object of class "logLik" with attributes for "nobs" and "df". The latter is equal to the number of estimated parameters plus 1 (the variance initialization value).

newsimpact *News Impact Curve*

Description

General method the news impact of a model

Usage

```
newsimpact(object, epsilon = NULL, ...)

## S3 method for class 'tsgarch.estimate'
newsimpact(object, epsilon = NULL, ...)
```

Arguments

object	an object of class "tsgarch.estimate".
epsilon	a user supplied zero mean noise vector. If this is NULL then a vector is created from the actual data using the minimum and maximum range.
...	additional parameters passed to the method.

Value

An object of class "tsgarch.newsimpact".

Note

The method does not support higher order GARCH models.

`nikkei` *Japanese NIKKEI Stock Index*

Description

The daily log returns in percent of the NIKKEI stock index spanning the period 1984-01-04 to 2000-12-22. In the original dataset there was a duplicate date on 2000-08-31 (but with a different value). Therefore, in order to correct this we have moved up the duplicate 2000-08-31 to become 2000-09-01, and the 2000-09-01 to 2000-09-02. Since the next date after this was 2000-09-04, no further adjustments were made. These changes preserve the original data in the order they appeared, with a minimal adjustment only to the index which has no impact on estimation, but avoiding internal warnings which arise on checks to the index. This dataset is included as it is used for the APARCH benchmark.

Usage

```
nikkei
```

Format

`nikkei`**:**

A data.frame containing 4246 observations in 2 columns:

index The string date in YYYY-MM-DD format.

value The daily log returns

Source

Journal of Applied Econometrics Data Archive 2003-v18.6/giot-laurent from the paper "Value-at-Risk for Long and Short Trading Positions" by Giot, Pierre and Sebastien Laurent, 2003, *Journal of Applied Econometrics*, 18(6), pp. 641–664.

`nloptr_fast_options` *Default options for nloptr solver*

Description

Default options for nloptr solver

Usage

```
nloptr_fast_options(
  trace = FALSE,
  xtol_rel = 1e-14,
  maxeval = 1000,
  xtol_abs = 1e-12
)

nloptr_global_options(
  trace = FALSE,
  xtol_rel = 1e-14,
  maxeval = 1000,
  xtol_abs = 1e-12
)
```

Arguments

`trace`	equivalent to option "print_level" for nloptr. High values results in more details.
`xtol_rel`	Stop when an optimization step (or an estimate of the optimum) changes every parameter by less than xtol_rel multiplied by the absolute value of the parameter.
`maxeval`	top when the number of function evaluations exceeds maxeval. This is not a strict maximum: the number of function evaluations may exceed maxeval slightly, depending upon the algorithm
`xtol_abs`	is a vector of length n (the number of elements in x) giving the tolerances: stop when an optimization step (or an estimate of the optimum) changes every parameter x(i) by less than xtol_abs(i).

Details

These as just a set of pre-created defaults which work well, particularly the "nloptr_fast_options" which uses an SQP solver. nloptr has many other solvers and combination of solvers which can be used. However, keep in mind that the solver must accept analytic derivatives as well as nonlinear inequality constraints.

Value

A list with options which can be passed to the solver.

nobs.tsgarch.estimate *Extract the Number of Observations*

Description

Extract the number of observations from an estimated model. This is principally intended to be used in computing BIC and used in other tidy methods

Usage

```
## S3 method for class 'tsgarch.estimate'
nobs(object, ...)
```

Arguments

object	an object of class "tsgarch.estimate".
...	not currently used.

Value

A numeric value.

omega *Omega (Variance Equation Intercept)*

Description

Returns the intercept of a GARCH model.

Usage

```
omega(object, ...)

## S3 method for class 'tsgarch.estimate'
omega(object, ...)

## S3 method for class 'tsgarch.spec'
omega(object, ...)
```

Arguments

object	an object.
...	additional parameters passed to the method.

Details

The intercept is either estimated directly as part of the model else indirectly if variance targeting was selected.

Value

a numeric value representing the value of the intercept.

---

persistence *Model Persistence*

Description

General method the persistence of a model.

Usage

```
persistence(object, ...)

## S3 method for class 'tsgarch.estimate'
persistence(object, ...)

## S3 method for class 'tsgarch.spec'
persistence(object, ...)
```

Arguments

object	an object of some class.
...	additional parameters passed to the method.

Value

The persistence of the estimated model. For GARCH models, the formulation varies by the type of model. See the vignette for more details.

pit.tsgarch.estimate *Probability Integral Transform (PIT)*

Description

Calculates and returns the conditional probability integral transform given the data and estimated density

Usage

```
## S3 method for class 'tsgarch.estimate'
pit(object, ...)
```

Arguments

object	an object.
...	not currently used.

Details

The PIT is essentially the probabilities returned from the cumulative distribution function (*p) given the data and estimated value of the mean, conditional standard deviation and any other distributional parameters.

Value

An xts vector of the conditional probabilities.

plot.tsgarch.estimate *Estimated Model Plots*

Description

Plot method for "tsgarch.estimate" class.

Usage

```
## S3 method for class 'tsgarch.estimate'
plot(x, y = NULL, ...)
```

Arguments

x	an object of class "tsgarch.estimate".
y	not used.
...	not used.

Value

a panel with plots for the estimated sigma value, the news impact curve and a "QQ" plot of the standardized residuals.

`plot.tsgarch.newsimpact` *News Impact Plot*

Description

Plot method for newsimpact class.

Usage

```
## S3 method for class 'tsgarch.newsimpact'
plot(x, y = NULL, ...)
```

Arguments

`x`	an object of class "tsgarch.newsimpact".
`y`	not used.
`...`	additional arguments pass to `plot.xy` other than "xlab", "ylab" and "main".

Value

a plot of the newsimpact curve

`predict.tsgarch.estimate` *Model Prediction*

Description

Prediction function for class "tsgarch.estimate".

Usage

```
## S3 method for class 'tsgarch.estimate'
predict(
  object,
  h = 1,
  newxreg = NULL,
  newvreg = NULL,
  nsim = 0,
  sim_method = c("parametric", "bootstrap"),
```

```
  block = 1,
  forc_dates = NULL,
  init_states = NULL,
  seed = NULL,
  ...
)
```

Arguments

object	an object of class "tsgarch.estimate".
h	the forecast horizon.
newxreg	not currently used,
newvreg	variance regressors rows equal to h. This can be either a numeric or xts matrix. Only needed if the model was estimated with regressors in the variance equation.
nsim	the number of simulations to use for generating the simulated predictive distribution. Defaults to zero (no simulated distribution).
sim_method	the simulation method to use when **nsim** great than zero. The "parametric" method samples from the model distribution whilst the "bootstrap" from the standardized model residuals.
block	for the "bootstrap" **sim_method**, this allows to generate block length samples (defaults to 1).
forc_dates	an optional vector of forecast dates equal to h. If NULL will use the implied periodicity of the data to generate a regular sequence of dates after the last available date in the data.
init_states	an optional vector of states to initialize the forecast. If NULL, will use the last available state from the estimated model. This must be equal to the max of the ARCH and GARCH terms.
seed	an integer that will be used in a call to set.seed before simulating.
...	additional arguments for future expansion options.

Details

The bootstrap method considered here, is based on re-sampling innovations from the empirical distribution of the fitted GARCH model to generate future realizations of the series and sigma. This only considers distributional uncertainty and will not generate prediction intervals for the 1-step ahead sigma forecast for which only the parameter uncertainty is relevant in GARCH type models (and not currently implemented). When the horizon **h** is equal to 1, no simulation is performaed since there is no uncertainty to account for.

Value

A "tsgarch.predict" object.

References

Pascual,L., Romo,J., Ruiz,E. (2006). "Bootstrap prediction for returns and volatilities in GARCH models." *Computational Statistics & Data Analysis*, **50**(9), 2293–2312.

print.summary.tsgarch.estimate

Model Estimation Summary Print method

Description

Print method for class "summary.tsgarch.estimate"

Usage

```
## S3 method for class 'summary.tsgarch.estimate'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	an object of class "summary.tsgarch.estimate".
digits	integer, used for number formatting. Optionally, to avoid scientific notation, set 'options(scipen=999)'.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.
...	not currently used.

Value

Invisibly returns the original summary object.

print.summary.tsgarch.profile

Profile Summary Print method

Description

Print method for class "summary.tsgarch.profile"

Usage

```
## S3 method for class 'summary.tsgarch.profile'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	an object of class "summary.tsgarch.estimate.profile".
digits	integer, used for number formatting. Optionally, to avoid scientific notation, set 'options(scipen=999)'.
...	not currently used.

Value

Invisibly returns the original summary object.

---

residuals.tsgarch.estimate

Extract Model Residuals

Description

Extract the residuals of the estimated model.

Usage

```
## S3 method for class 'tsgarch.estimate'
residuals(object, standardize = FALSE, ...)

## S3 method for class 'tsgarch.multi_estimate'
residuals(object, standardize = FALSE, ...)
```

Arguments

object	an object of class "tsgarch.estimate" or "tsgarch.multi_estimate".
standardize	logical. Whether to standardize the residuals by the conditional volatility.
...	not currently used.

Value

An xts vector of the model residuals for the univariate type objects. In the case of a multi-estimate object, a list of xts vectors is returned if the individual univariate objects have unequal indices, else an xts matrix is returned. Note that If the model had no constant in the conditional mean equation then this just returns the original data (which is assumed to be zero mean noise).

sigma.tsgarch.estimate *Extract Volatility (Conditional Standard Deviation)*

Description

Extract the conditional standard deviation from a GARCH model.

Usage

```
## S3 method for class 'tsgarch.estimate'
sigma(object, ...)

## S3 method for class 'tsgarch.multi_estimate'
sigma(object, ...)
```

Arguments

object	an object of class "tsgarch.estimate", "tsgarch.predict", "tsgarch.simulate" or a "tsgarch.multi_estimate".
...	not currently used.

Value

An xts vector of the conditional volatility for the univariate type objects. In the case of a multi-estimate object, a list of xts vectors is returned if the individual univariate objects have unequal indices, else an xts matrix is returned.

simulate.tsgarch.spec *Model Simulation*

Description

Simulates paths of a GARCH model.

Usage

```
## S3 method for class 'tsgarch.spec'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  h = 1000,
  var_init = NULL,
  innov = NULL,
```

```
  innov_init = NULL,
  vreg = NULL,
  burn = 0,
  ...
)
```

Arguments

object	an object of class "tsgarch.spec".
nsim	the number of sample paths to generate.
seed	an integer that will be used in a call to set.seed before simulating.
h	the number of time steps to simulate paths for.
var_init	the seed value for initializing the variance equation recursion. If NULL, the variance target value is used based on the supplied parameters. This should be a vector and assumes all sample paths are seeded the same way.
innov	an optional matrix of dimensions nsim by h of zero mean unit variance (standardized) innovations which will be used instead of the model distribution for simulation. No checks are performed on whether the supplied values are standardized.
innov_init	an optional vector of initialization values for the standardized innovations. This allows the seeding of the initial innovations with user supplied values (useful when simulating forward from an existing model for the purpose of continuing the modeled series from some fixed point).
vreg	an optional vector of length h representing any pre-multiplied variance regressors to use in the simulation.
burn	burn in. Will be discarded before returning the output.
...	for aparch, fgarch, egarch and gjrgarch models, an optional vector of length max(q,p) with values for initializing the ARCH equation and named "arch_initial". This is mostly used for validation purposes. The "arch_initial" value is always returned by an estimated object.

Details

Once a GARCH model is specified via garch_modelspec, the slot "parmatrix" contains initial values for the parameters which can be used to set them to any value for the simulation. This matrix as well as details of the model (type, order, distribution) are the only pieces of information used in the simulation. The "vreg" argument in the spec will be ignored. Instead, the user can supply a pre-multiplied vector to the simulate function which will be used. Note that the "multiplicative" argument in the specification will be used in this case to determine how the regressors enter the conditional variance equation. While the "innov" argument must be a matrix, all other values are vectors and assume that they will be the same across all sample paths. If the user wants to assign different values for arguments "var_init", "innov_init" and "vreg", then the simulate method should be called multiple times.

Value

An object of class "tsgarch.simulate" with slots for the simulated sigma and series simulated distributions which are each of class "tsmodel.distribution". The simulated error (not returned) is equal to the simulated series less the mean equation constant if not equal to zero.

`summary.tsgarch.estimate`
GARCH Model Estimation Summary

Description

Summary method for class "tsgarch.estimate"

Usage

```
## S3 method for class 'tsgarch.estimate'
summary(object, digits = 4, vcov_type = "H", include_persistence = TRUE, ...)
```

Arguments

`object`	an object of class "tsgarch.estimate".
`digits`	integer, used for number formatting. Optionally, to avoid scientific notation, set 'options(scipen=999)'.
`vcov_type`	the type of standard errors based on the vcov estimate (see `vcov`).
`include_persistence`	whether to include the estimate of the persistence and its calculated standard errors (calculated using the `sdreport`) in the output.
`...`	not currently used.

Value

A list with summary information of class "summary.tsgarch.estimate".

`summary.tsgarch.profile`
GARCH Profile Summary

Description

Summary method for class "tsgarch.profile"

Usage

```
## S3 method for class 'tsgarch.profile'
summary(object, digits = 4, measure = "RMSE", ...)
```

Arguments

object	an object of class "tsgarch.profile".
digits	integer, used for number formatting. Optionally, to avoid scientific notation, set 'options(scipen=999)'.
measure	either one of the included measure in the summary slot of the returned object, which currently includes the relative error measures "RMSE", "MAE", "MAPE", summary measures on the estimated values "MEAN", "MEDIAN", "P20" and "P80", else any other user calculated measure which has been generated in the summary table post processing.
...	not currently used.

Value

A list with summary information of class "summary.tsgarch.profile".

---

to_multi_estimate *Convert a list of tsgarch.estimate objects to a multi_estimate object*

Description

Convert a list of tsgarch.estimate objects to a multi_estimate object

Usage

```
to_multi_estimate(object, ...)
```

Arguments

object	an list with "tsgarch.estimate" objects.
...	none

Details

This is a convenience method which provides the flexibility to manually estimate each series and then convert the list of these objects to a multi-estimation object, rather than having to use the estimate method on a multi-specification object.

Value

a validated object of class "tsgarch.multi_estimate".

Author(s)

Alexios Galanos

`tsbacktest.tsgarch.spec` *Walk Forward Rolling Backtest*

Description

Generates an expanding window walk forward backtest with option for rolling the forecast by filtering (see details).

Usage

```
## S3 method for class 'tsgarch.spec'
tsbacktest(
  object,
  start = floor(length(object$target$y_orig))/2,
  end = length(object$target$y_orig),
  h = 1,
  estimate_every = 1,
  rolling = FALSE,
  trace = FALSE,
  ...
)
```

Arguments

`object`	an object of class "tsgarch.spec".
`start`	numeric data index from which to start the backtest.
`end`	numeric data index on which to end the backtest. The backtest will end 1 period before that date in order to have at least 1 out of sample value to compare against.
`h`	forecast horizon. As the expanding window approaches the "end", the horizon will automatically shrink to the number of available out of sample periods.
`estimate_every`	number of periods at which the model is re-estimated (defaults to 1).
`rolling`	this indicates whether forecasts are made only on the estimation date (FALSE) or whether to filter the data 1 period at a time and forecast from the filtered data (TRUE).
`trace`	whether to show the progress bar. The user is expected to have set up appropriate handlers for this using the "progressr" package.
`...`	not currently used.

Details

The rolling option allows to re-estimate the data every n periods whilst filtering the data 1-step ahead between re-estimation dates so that overlapping forecasts are generated.

Value

A list which includes a data.table having the following columns:

- estimation_date: the date at which the model was estimated.
- convergence: whether both kkt1 and kkt2 were TRUE (Kuhn Karush Tucker conditions) from the kktchk function in optimx.
- filter_date: the date on which a prediction was generated. For rolling prediction this means that an estimated model was filtered for new data prior to re-predicting.
- horizon: the forecast horizon of the prediction.
- size: the length of the data used in estimation.
- forecast_date: the date corresponding to the forecast.
- mu: the conditional mean prediction.
- sigma: the conditional volatility prediction.
- skew: the distribution skew parameter (non-time varying hence constant across each estimation window).
- shape: the distribution shape parameter (non-time varying hence constant across each estimation window).
- shape: the distribution lambda parameter (non-time varying hence constant across each estimation window).
- actual: the actual observation corresponding to the forecast date.

Additional slots in the list include the distribution used and other information relating to the backtest setup.

Note

The function can use parallel functionality as long as the user has set up a `plan` using the future package.

`tsequation.tsgarch.estimate`

Model Equation (LaTeX)

Description

Generates a list of model equations in LaTeX.

Usage

```
## S3 method for class 'tsgarch.estimate'
tsequation(object, ...)
```

Arguments

object	an object of class "tsgarch.estimate".
...	not currently used.

Details

This method is called in the summary when the format output option chosen is "flextable".

Value

A list of equations in LaTeX which can be used in documents. This is a list with 4 slots for the conditional distribution, the conditional volatility, the persistence and unconditional variance equations.

---

tsfilter.tsgarch.estimate

Model Filtering

Description

Filters new data based on an already estimated model or filters data based on a specification object.

Usage

```
## S3 method for class 'tsgarch.estimate'
tsfilter(object, y = NULL, newxreg = NULL, newvreg = NULL, ...)

## S3 method for class 'tsgarch.spec'
tsfilter(object, y = NULL, newxreg = NULL, newvreg = NULL, ...)
```

Arguments

object	an object of class "tsgarch.estimate" or "tsgarch.spec".
y	an xts vector of new values to filter. Can also be NULL in which case the
newxreg	not currently used,
newvreg	variance regressors with the same number of rows as y. This can be either a numeric or xts matrix. Only needed if the model was estimated with regressors in the variance equation.
...	additional arguments for future expansion.

Details

The method filters new data and updates the object with this new information so that it can be called recursively as new data arrives. It is also possible to use a specification object with fixed parameters, by appropriately setting the values of the "parmatrix" object in the specification slot. In this case, the returned object will also be of classs "tsgarch.estimate". If an object of "tsgarch.spec" is used with y not NULL, then the method will first filter the values of the data in the object, generating an object of "tsgarch.estimate" and then call the method again on this new object and the new y values (and optionally any newvreg values). In this way, using either object classes will return the exact same results. The timestamp indices of y must be strictly greater than the maximum timestamp index of the data within the object (i.e. we only filter on new data).

Value

A "tsgarch.estimate" object with updated information.

---

tsprofile.tsgarch.spec

Model Parameter Profiling

Description

Profiles the model parameters under the assumptions of the model.

Usage

```
## S3 method for class 'tsgarch.spec'
tsprofile(
  object,
  nsim = 100,
  sizes = c(800, 1000, 1500, 2000, 3000),
  var_init = NULL,
  seed = NULL,
  burn = 0,
  trace = FALSE,
  ...
)
```

Arguments

object	an object of class "tsgarch.spec" with pre-set parameters.
nsim	the number of paths to generate.
sizes	a vector of data sizes for which to simulate and estimate.
var_init	the variance to use to initialize the simulation.
seed	an object specifying if and how the random number generator should be initialized. See the simulate documentation for more details.

burn	burn in samples.
trace	whether to show the progress bar. The user is expected to have set up appropriate handlers for this using the "progressr" package.
...	not currently used.

Details

The function profiles the parameters of a model by simulating and then estimating multiple paths from the assumed DGP. This makes it possible to obtain a better understanding of the convergence properties (RMSE) of each parameter under different data sizes.

Value

An object of class "tsgarch.profile".

Note

The function can use parallel functionality as long as the user has set up a `plan` using the future package. External regressors are not supported at this time and an error will occur if persent in the specification.

---

unconditional.tsgarch.estimate

Unconditional Value

Description

Unconditional value of a GARCH model variance.

Usage

```
## S3 method for class 'tsgarch.estimate'
unconditional(object, ...)

## S3 method for class 'tsgarch.spec'
unconditional(object, ...)
```

Arguments

object	an object of class "tsgarch.estimate" or "tsgarch.spec".
...	not currently used.

Details

For some models, there is no closed form solution available for the unconditional variance of higher order model (e.g. GARCH(2,1)) in which case a simulation based approach is adopted to approximate the value.

Value

A numeric vector of length 1 of the unconditional variance of the model.

`vcov.tsgarch.estimate` *The Covariance Matrix of the Estimated Parameters*

Description

The Covariance Matrix of the Estimated Parameters

Usage

```
## S3 method for class 'tsgarch.estimate'
vcov(object, adjust = FALSE, type = c("H", "OP", "QMLE", "NW"), ...)
```

Arguments

`object`	an object of class tsgarch.estimate.
`adjust`	logical. Should a finite sample adjustment be made? This amounts to multiplication with n/(n-k) where n is the number of observations and k the number of estimated parameters.
`type`	valid choices are "H" for using the analytic hessian for the bread, "OP" for the outer product of gradients, "QMLE" for the Quasi-ML sandwich estimator (Huber-White), and "NW" for the Newey-West adjusted sandwich estimator (a HAC estimator).
`...`	additional parameters passed to the Newey-West bandwidth function to determine the optimal lags.

Value

The variance-covariance matrix of the estimated parameters.

Index