Package 'MSGARCH'

January 19, 2017

Type Package

Title Markov-Switching GARCH Models

Version 0.17.8

Date 2016-01-19

Author David Ardia [aut],
Keven Bluteau [aut, cre],
Kris Boudt [ctb],
Brian Peterson [ctb],
Denis-Alexandre Trottier [aut]

Maintainer Keven Bluteau <Keven.Bluteau@unine.ch>

Description The MSGARCH package offers methods to fit (by Maximum Likelihood or Bayesian), simulate, and forecast various Markov-Switching GARCH processes.

License GPL (>= 2)

Imports Rcpp, adaptMCMC, nloptr, DEoptim, methods, stringr, ggplot2, reshape2, zoo, expm, fanplot, dfoptim

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 5.0.1

NeedsCompilation yes

R topics documented:

MSGARCH-package *The R package MSGARCH*

Description

The R package MSGARCH aims to provide a comprehensive set of functionalities for Markov-switching GARCH processes (Haas et al. 2004), including fitting, filtering, forecasting, and simulating. Other functions related to Value-at-Risk, Expected-Shortfall, and conditional distributions are also available. The main functions of the package are coded in C++ with Rcpp (Eddelbuettel and Francois, 2011) and RcppArmadillo (Eddelbuettel and Sanderson, 2014). In the R package MSGARCH there is no equation for the mean as in the R package rugarch (Ghalanos, 2015). This means that we assume that before modeling, the user has filtered the mean from their time series.
We provide a variety of single-regime GARCH processes and regime-switching allowing for many conditional distributions. This allows for a rich modeling environment for Markov-switching GARCH models. Each single-regime process is a one-lag process (e.g., GARCH(1,1)). Allowing for only one-lag has proved to be sufficient in many cases and it reduces models complexity which can become a problem during the optimization procedure. When optimization is taking place, we ensure that each regime is covariance-stationary and strictly positive (see details in kernel for more information) which makes the entire process also covariance-stationary and strickly positive. We also set a condition that each unique single-regime models type in a multiple-regime framework are in order of unconditional volatility. This means that is if a three regimes specification with two sGARCH regimes and one gjrGARCH regime is constructed with create.spec, the first sGARCH regime will have a lower unconditional volatility than the second sGARCH regime while the gjrGARCH regime can have any unconditional volatility since it is the only regime with this model. For a full demonstration of the package please read Markov-Switching GARCH Models in R: The MSGARCH Package (see https://ssrn.com/abstract=2845809). The authors acknowledge Google for financial support via the Google Summer of Code 2016 project "MSGARCH"; see https://summerofcode.withgoogle.com/projects/#6497774455488512 , the International Institute of Forecasting and Industrielle-Alliance.

References

Ardia, D. and Bluteau, K. and Boudt, K. and Trottier, D.-A. (2016). Markov-Switching GARCH Models in R: The MSGARCH Package. https://ssrn.com/abstract=2845809

Eddelbuettel, D. Francois, R. (2011). Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, 40, pp. 1-18, http://www.jstatsoft.org/v40/i08/.

Eddelbuettel, D. Sanderson, C. (2014). RcppArmadillo: Accelerating R with High–Performance C++ Linear Algebra. *Computational Statistics & Data Analysis*, 71, pp. 1054-1063, http://dx.doi.org/10.1016/j.csda.2013.02.005.

Haas, M. Mittnik, S. Paolella, MS. (2004). A New Approach to Markov-Switching GARCH Models. *Journal of Financial Econometrics*, 2, pp. 493-530, http://doi.org/10.1093/jjfinec/nbh020

Ghalanos, A. (2015). rugarch: Univariate GARCH Models. https://cran.r-project.org/package=rugarch.

AIC *Compute Akaike information criterion (AIC).*

Description

Compute Akaike information criterion (AIC).

Usage

```
AIC(fit)
```

Arguments

fit	Fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.

Details

Compute Akaike information criterion (AIC) based on the work of Akaike (Akaike, 1974). If a matrix of MCMC posterior draws estimates is given, the AIC on the posterior mean is calculated.

Value

AIC value.

References

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19, pp. 716-723.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model by MLE
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# compute AIC
AIC = MSGARCH::AIC(fit)
```

AMZN *Log return of Amazon inc. closing Value*

Description

The Amazon inc. closing value log return from 1998-01-01 to 2015-12-31 from Yahoo Finance https://finance.yahoo.com/.

Usage

```
data("AMZN")
```

Format

Matrix containing 4,529 observations.

Source

Yahoo Finance https://finance.yahoo.com/

BIC *Compute Bayesian information criterion (BIC).*

Description

Compute Bayesian information criterion (BIC).

Usage

```
BIC(fit)
```

Arguments

fit	Fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes

Details

Compute Bayesian information criterion (BIC) based on the work of Schwarz (Schwarz, 1978). If a matrix of MCMC posterior draws estimates is given, the BIC on the posterior mean is calculated.

Value

BIC value.

References

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6, pp. 461-464.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model by MLE
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# compute BIC
BIC = MSGARCH::BIC(fit)
```

---

cdf *Cumulative distribution function.*

Description

Method returning the cumulative distribution function in-sample or of a vector of points consider as one step ahead draws (`t = T + 1`).

Usage

```
cdf(object, x, theta, y, log = FALSE, do.its = FALSE)
```

Arguments

`object`	Model specification of class `MSGARCH_SPEC` created with `create.spec` or fit object of type `MSGARCH_MLE_FIT` created with `fit.mle` or `MSGARCH_BAY_FIT` created with `fit.bayes`.
`x`	Vector (of size N) of points evaluated at `t = T + 1` (used when `do.its = FALSE`).
`theta`	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
`y`	Vector (of size T) of observations (not required when using a fit object).
`log`	Boolean indicating if the log cumulative is returned. (Default: `log = FALSE`)
`do.its`	Boolean indicating if the in-sample cdf is returned. (Default: `do.its = FALSE`)

Details

If a matrix of parameter estimates is given, each parameter estimate (each row) is evaluated individually. If `do.its = FALSE`, the points `x` are evaluated as `t = T + 1` realizations and the method uses the variance estimate at `t = T + 1`. If `do.its = TRUE`, `y` is evaluated using their respective variance estimate at each time `t`.

Value

A list of class MSGARCH_CDF containing three components:

- cdf:
  If do.its = FALSE: (Log-)Cumulative of the points x at t = T + 1 (vector of size N or matrix of size M x N).
  If do.its = TRUE: In-sample (Log-)Cumulative of y (vector of size T or matrix of size M x T).
- x:
  If do.its = FALSE: Vector (of size N) of points evaluated at t = T + 1.
  If do.its = TRUE: Vector (of size T) of observations.
- do.its: Orinigal user inputed do.its for reference.

The class MSGARCH_CDF contains the plot method only if do.its = FALSE.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# run pdf method in-sample
cdf.its = MSGARCH::cdf(object = fit, log = FALSE, do.its = TRUE)

# create mesh
x = seq(-3,3,0.01)

# run cdf method on mesh at T + 1
cdf = MSGARCH::cdf(object = fit, x = x, log = FALSE, do.its = FALSE)

plot(cdf)
```

---

create.spec *Model specification*

Description

Function for creating a model specification before fitting and using the R package MSGARCH functionalities.

Usage

```
create.spec(model = c("sGARCH", "sGARCH"), distribution = c("norm", "norm"),
  do.skew = c(FALSE, FALSE), do.mix = FALSE, do.shape.ind = FALSE)
```

Arguments

model	Vector (of size K) containing the variance model specifications. Valid models are "sGARCH", "eGARCH", "gjrGARCH", "tGARCH", and "GAS". (Default: model = c("sGARCH", "sGARCH"))
distribution	Vector (of size K) of conditional densities. Valid distributions are "norm", "std", and "ged". The vector must be of the same length as the models' vector. (Default: distribution = c("norm", "norm"))
do.skew	Vector (of size K) of boolean indicating if the conditional density is skewed. The vector must be of the same length as the distributions' vector. (Default: do.skew = c(FALSE, FALSE))
do.mix	Boolean indicating if the specification is a mixture type. If the argument is TRUE, a Mixture of GARCH is created, while if the argument is FALSE, a Markov-Switching GARCH is created (see details). (Default: do.mix = FALSE)
do.shape.ind	Boolean indicating if the distribution are Regime-Independent. If the argument is TRUE, all distributions are the same and the distribution parameters do not depend on the regime to which the distribution is attributed. If the argument is TRUE, all distributions in the distribution argument and all skew arguments must be the same. (Default: do.shape.ind = FALSE)

Details

The Markov-Switching specification created is based on the Haas et al. (2004a) MSGARCH specification. It is a MSGARCH model that is separated in K single-regimes specifications which are updated in parallel. Under the Haas et al. (2004a) specification, the conditional variance is a function of the past data and the current state. The Mixture of GARCH option is based on Haas et al. (2004b). A Mixture of GARCH is a mixture of distributions where the variance process of each distribution is a single-regime process. Every single-regime specification is a one-lag process (e.g., GARCH(1,1)) since it has proved to be sufficient in financial econometrics. This simplification of the processes also reduces models' complexity which can become a problem during the optimization procedure.

Value

A list of class MSGARCH_SPEC containing variables related to the created specification.
The list contains:

- theta0 : Vector (of size d) of default parameters.
- is.mix : Boolean indicating if the specification is a mixture.
- is.shape.ind : Boolean indicating if the distributions' parameters are regime-independent.
- K : Number of regimes.
- sigma0 : Default variance-covariance matrix (of size K x K) used for the Bayesian esimation.
- lower : Vector (of size d) of lower parameters' bounds.
- upper : Vector (of size d) of upper parameters' bounds.
- ineqlb : Vector (of size d) of lower inequality bounds.
- inequb : Vector (of size d) of upper inequality bounds.
- n.params : Vector (of size K) of the total number of parameters by regime including distributions' parameters.

- `n.params.vol` : Vector (of size K) of the total number of parameters by regime excluding distributions' parameters.
- `do.init` : Boolean indicating the default `do.init` argument.
- `label` : Vector (of size d) of parameters' labels.
- `name` : Vector (of size K) of model specifications' names.
- `func` : List of internally used R functions.
- `rcpp.func` : List of internally used `Rcpp` functions.

The `MSGARCH_SPEC` class possesses these methods:

- `sim` : Simulation method.
- `simahead` : Step ahead simulation method.
- `ht` : Conditional volatility in each regime.
- `kernel` : Kernel method.
- `unc.vol` : Unconditional volatility in each regime.
- `pred` : Predictive method.
- `pit` : Probability Integral Transform.
- `risk` : Value-at-Risk And Expected-Shortfall methods.
- `pdf` : Probability density function.
- `cdf` : Cumulative distribution function.
- `Pstate` : State probabilities filtering method.
- `fit.mle` : Maximum Likelihood estimation.
- `fit.bayes` : Bayesian estimation.
- `print` and `summary` : Summary of the created specification.

References

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31, pp. 307-327.

Creal, D. Koopman, S. J. & Lucas, A. (2013). Generalized Autoregressive Score Models with Applications. *Journal of Applied Econometrics*, 28, pp. 777-795.

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

Glosten, L. R. Jagannathan, R. & Runkle, D. E. (1993). On the Relation Between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *Journal of Finance*, 48, pp. 1779-1801.

Haas, M. Mittnik, S. & Paolella, M. S. (2004a). A New Approach to Markov-Switching GARCH Models. *Journal of Financial Econometrics*, 2, pp. 493-530.

Haas, M. Mittnik, S. & Paolella, M. S. (2004b). Mixed Normal Conditional Heteroskedasticity. *Journal of Financial Econometrics*, 2, pp. 211-250.

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

Zakoian, J.-M. (1994). Threshold Heteroskedastic Models. *Journal of Economic Dynamics and Control*, 18, pp. 931-955.

Examples

```
# create model specification
spec = MSGARCH::create.spec(model = c("sGARCH","gjrGARCH"), distribution = c("norm","std"),
                            do.skew = c(TRUE,FALSE), do.mix = FALSE, do.shape.ind = FALSE)
print(spec)
```

---

crps *CRPS (continuous ranked probability score) measure.*

Description

Method returning the CRPS at `t = T + 1`.

Usage

```
crps(object, yn, ctr = list(lower = -20, upper = 20, n.mesh = 500, a = 0, b =
  1))
```

Arguments

`object`	Model specification of class `MSGARCH_SPEC` created with `create.spec` or fit object of type `MSGARCH_MLE_FIT` created with `fit.mle` or `MSGARCH_BAY_FIT` created with `fit.bayes`.
`yn`	Scalar value to be evaluated at `t = T + 1`.
`ctr`	Control list parameters.

Details

If a matrix of MCMC posterior draws estimates is given, the Bayesian CRPS is calculated.

Value

A vector with five crps measures

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# run at T + 1 from model
crps = MSGARCH::crps(object = fit, yn = 0.6)
```

DIC *Compute Deviance Information Criterion (DIC).*

Description

Compute Deviance Information Criterion (DIC).

Usage

```
DIC(fit)
```

Arguments

fit	Fit object of type MSGARCH_BAY_FIT created with fit.bayes.

Details

Compute the Deviance information criterion of Spiegelhalter, David J., et al. (2002). We define the deviance as:

$$D(\theta) = -2LLH(\mathbf{y}|\theta),$$

where $\mathbf{y}$ are the data, θ are the parameters, and LLH() is the log-likelihood function. The expectation

$$\bar{D} = \mathbf{E}^{\theta}[D(\theta)],$$

where $\mathbf{E}^{\theta}$ is the expectation over all theta in a MCMC chain, is a measure of how well the model fits the data. The larger this expectation is, the worse is the fit. The effective number of parameters of the model can be defined as

$$p_V = \frac{1}{2}\widehat{var}\left(D(\theta)\right),$$

where $\widehat{var}$ is the the population variance estimator. The larger the effective number of parameters is, the easier it is for the model to fit the data, and so the deviance needs to be penalized. Finally DIC is defined as:

$$DIC = p_V + \bar{D}.$$

Value

A list containing four variables:

- DIC : Deviance Information Criterion.
- IC : Bayesian Predictive Information Criterion (IC = 2 * pV + D.bar).
- pV : Effective number of parameters (pV = var(D)/2)
- D.bar: Expected value of the deviance over the posterior

References

Spiegelhalter, David J., et al. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model by Bayesian estimation
set.seed(123)
fit = MSGARCH::fit.bayes(spec = spec, y = sp500)

# compute DIC
DIC = MSGARCH::DIC(fit)
```

`fit.bayes` *Bayesian estimation.*

Description

Method that performs Bayesian estimation of a `MSGARCH_SPEC` object on a set of observations.

Usage

```
fit.bayes(spec, y, ctr = list())
```

Arguments

`spec`	Model specification of class `MSGARCH_SPEC` created with `create.spec`.
`y`	Vector (of size T) of observations.
`ctr`	A list of control parameters. The control parameters have three components: • `N.burn` (integer >= 0): Number of discarded draws. (Default: `N.burn = 5000`) • `N.mcmc` (integer > 0) : Number of draws. (Default: `N.mcmc = 10000`) • `N.thin` (integer > 0) : Thinning factor (every `N.thin` draws are kept). (Default: `N.thin = 10`) • `theta0` : Starting value for the chain (if empty the specification default values are used). • `do.enhance.theta0` : Boolean indicating if the default parameters values are enhanced using `y` variance. (Default: `do.enhance.theta0 = FALSE`)

Details

The total number of draws is equal to `N.mcmc / N.thin`. The Bayesian estimation uses the R package `adaptMCMC` (Andreas, 2012) which implements the adaptive sampler of Vihola (2012). The starting parameters are the specification default parameters. The argument `do.enhance.theta0` uses the volatilities of rolling windows of `y` and adjusts the default parameter of the specification so that the unconditional volatility of each regime is set to different quantiles of the volatilities of the rolling windows of `y`.

Value

A list of class MSGARCH_BAY_FIT containing four components:

- theta : The MCMC chain (matrix from the R package coda (Plummer et al., 2006) of size N.mcmc / N.thin).
- accept : Acceptation rate of the sampler.
- y : Vector (of size T) of observations.
- spec : Model specification of class MSGARCH_SPEC created with create.spec.

The MSGARCH_BAY_FIT contains these methods:

- AIC : Compute Akaike information criterion (AIC).
- BIC : Compute Bayesian information criterion (BIC).
- DIC : Compute Deviance Information Criterion (DIC).
- ht : Conditional volatility in each regime.
- kernel : Kernel method.
- unc.vol : Unconditional volatility in each regime.
- pred : Predictive method.
- pit : Probability Integral Transform.
- risk : Value-at-Risk And Expected-Shortfall methods.
- simahead : Step ahead simulation method.
- sim : Simulation method.
- pdf : Probability density function.
- cdf : Cumulative distribution function.
- Pstate : State probabilities filtering method.
- summary : Summary of the fit.

References

Andreas, S. (2012). adaptMCMC: Implementation of a Generic Adaptive Monte Carlo Markov Chain Sampler. https://cran.r-project.org/package=adaptMCMC.

Plummer, M. Best, N. Cowles, K. & Vines, K. (2006). CODA: Convergence Diagnosis and Output Analysis for MCMC. *R News*, 6, pp.7-11. https://cran.r-project.org/package=coda.

Vihola, M. (2012). Robust Adaptive Metropolis Algorithm with Coerced Acceptance Rate. *Statistics and Computing*, 22, pp. 997-1008.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with Bayesian estimation
set.seed(123)
```

```
fit = MSGARCH::fit.bayes(spec = spec, y = sp500,
                         ctr = list(N.burn = 500, N.mcmc = 1000, N.thin = 1))

summary(fit)
```

`fit.mle` *ML estimation.*

Description

Method that performs Maximum Likelihood estimation of a `MSGARCH_SPEC` object on a set of observations.

Usage

```
fit.mle(spec, y, ctr = list())
```

Arguments

`spec`	Model specification created with `create.spec`.
`y`	Vector (of size T) of observations.
`ctr`	List of control parameters. The control parameters have two components to it:

- `do.init` : Boolean indicating if there is a pre-optimization with the R package `DEoptim` (Ardia et al., 2011). (Default: `do.init = FALSE`)
- `NP` : Number of parameter vectors in the population in `DEoptim` optimization. (Default: `NP = 200`)
- `itermax` : Maximum iteration (population generation) allowed in `DEoptim` optimization. (Default: `maxit = 200`)
- `theta0` : Starting value for the chain (if empty the specification default value are used).
- `do.enhance.theta0` : Boolean indicating if the default parameters value are enhance using `y` variance. (Default: `do.enhance.theta0 = TRUE`)

Details

The Maximum likelihood estimation uses the R package `dfoptim` (Varadhan and Borchers, 2016) for main optimizer and `nloptr` (Johnson, 2014) in case of non-convergence while it uses the R package `DEoptim` when `do.init = TRUE` as an initialization for `dfoptim` and `nloptr`. The starting parameters are the specification default parameters. The argument `do.enhance.theta0` uses the volatilities of rolling windows of `y` and adjust the starting parameters of the specification so that the unconditional volatility of each regime is set to different quantiles of the volatilities of the rolling windows of `y`.

Value

A list of class `MSGARCH_MLE_FIT` containing five components:

- `theta` : Optimal parameters (vector of size d).
- `log_kernel` : log-kernel of `y` given the optimal parameters.
- `spec` : Model specification of class `MSGARCH_SPEC` created with `create.spec`.

- `is.init` : Indicating if estimation was made with do.init option.
- `y` : Vector (of size T) of observations.

The `MSGARCH_MLE_FIT` contains these methods:

- `AIC` : Compute Akaike information criterion (AIC).
- `BIC` : Compute Bayesian information criterion (BIC).
- `ht` : Conditional volatility in each regime.
- `kernel` : Kernel method.
- `unc.vol` : Unconditional volatility in each regime.
- `pred` : Predictive method.
- `pit` : Probability Integral Transform.
- `risk` : Value-at-Risk And Expected-Shortfall methods.
- `simahead` : Step ahead simulation method.
- `sim` : Simulation method.
- `pdf` : Probability density function.
- `cdf` : Cumulative distribution function.
- `Pstate` : State probabilities filtering method.
- `summary` : Summary of the fit.

References

Ardia, D. Boudt, K. Carl, P. Mullen, K. M. & Peterson, B. G. (2011). Differential Evolution with `DEoptim`. *R Journal*, 3, pp. 27-34

Ardia, D. Mullen, K. M. Peterson, B. G. & Ulrich, J. (2015). `DEoptim`: Differential Evolution in R. https://cran.r-project.org/package=DEoptim

Mullen, K. M. Ardia, D. Gil, D. L. Windover, D. Cline, J.(2011) `DEoptim`: An R Package for Global Optimization by Differential Evolution. *Journal of Statistical Software*, 40, pp. 1-26, DOI: http://dx.doi.org/10.18637/jss.v040.i06

Johnson, S. G. (2014). The NLopt Nonlinear-Optimization. https://cran.r-project.org/package=nloptr.

Varadhan, R., Borchers H. W. (2016. dfoptim: Derivative-Free Optimization. https://cran.r-project.org/package=dfoptim.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation using DEoptim intialization
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500)
summary(fit)
```

ht *Conditional variance in each regime.*

Description

Method returning the conditional variance of each regime.

Usage

```
ht(object, theta, y)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
y	Vector (of size T) of observations (not required when using a fit object).

Details

If a matrix of parameter estimates is given, each parameter estimate (each row) is evaluated individually.

Value

Condititional variance (array of size (T + 1) x M x K) in each regime.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500)

# Compute the conditional variance
ht = MSGARCH::ht(object = fit)

plot(ht)
```

`kernel` *Kernel function.*

Description

Method returning the kernel value of a vector of observations given a model specification.

Usage

```
kernel(object, theta, y, log = TRUE)
```

Arguments

`object`	Model specification of class `MSGARCH_SPEC` created with `create.spec` or fit object of type `MSGARCH_MLE_FIT` created with `fit.mle` or `MSGARCH_BAY_FIT` created with `fit.bayes`.
`theta`	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
`y`	Vector (of size T) of observations (not require when using a fit object).
`log`	Boolean indicating if the log kernel is returned. (Default: `log = TRUE`)

Details

If a matrix of parameter estimates is given, each parameter estimate (each row) is evaluated individually. The kernel is a combination of the prior and the likelihood function. The kernel is equal to prior(θ) + L(y|θ) where L is the likelihood of y given the parameter θ. When doing optimization, the goal is to minimize the negative log-kernel.

- Details on the prior
  The prior is different for each specification. It ensures that the θ makes the conditional variance process stationary, positive, and that it respectsthat the sums of the probabilities in the case of a multiple-regime model are all equal to 1. If any of these three conditions is not respected the prior returns `-1e10`, meaning that the optimizer or the sampler will know that θ is not a good candidate.

Value

(Log-)Kernel value (scalar or vector of size M) of the vector of observations.

References

Hamilton, J. D. (1989) A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle. *Econometrica*, 57, pp.357-38

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# compute the kernel
kernel = MSGARCH::kernel(fit, log = TRUE)
```

---

pdf *Probability density function.*

Description

Method returning the probability density in-sample or of a vector of points consider as one step ahead draws (`t = T + 1`).

Usage

```
pdf(object, x, theta, y, log = FALSE, do.its = FALSE)
```

Arguments

`object`	Model specification of class `MSGARCH_SPEC` created with `create.spec` or fit object of type `MSGARCH_MLE_FIT` created with `fit.mle` or `MSGARCH_BAY_FIT` created with `fit.bayes`.
`x`	Vector (of size N) of points evaluated at `t = T + 1` (used when `do.its = FALSE`).
`theta`	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
`y`	Vector (of size T) of observations (not required when using a fit object).
`log`	Boolean indicating if the log-density is returned. (Default: `log = FALSE`)
`do.its`	Boolean indicating if the in-sample pdf is returned. (Default: `do.its = FALSE`)

Details

If a matrix of parameter estimates is given, each parameter estimate (each row) is evaluated individually. If `do.its = FALSE`, the points `x` are evaluated as `t = T + 1` realization and the method uses the variance estimate at `t = T + 1`. If `do.its = TRUE`, `y` is evaluated using their respective variance estimate at each time `t`.

Value

A list of class MSGARCH_PDF containing three components:

- pdf:
  If do.its = FALSE: (Log-)Probability density of the points x at t = T + 1 (vector of size N or matrix of size M x N)
  If do.its = TRUE: In-sample (Log-)Probability density of y (vector of size T or matrix of size M x T).
- x:
  If do.its = FALSE: Vector (of size N) of points evaluated at t = T + 1.
  If do.its = TRUE: Vector (of size T) of observations.
- do.its: Orinigal user inputed do.its for reference.

The class MSGARCH_PDF contains the plot method only if do.its = FALSE.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation using DEoptim intialization
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# run pdf method in-sample
pdf.its = MSGARCH::pdf(object = fit, log = FALSE, do.its = TRUE)

sum(pdf.its$pdf, na.rm = TRUE)
# create mesh

x = seq(-3,3,0.01)

# run pdf method on mesh at T + 1
pdf = MSGARCH::pdf(object = fit, x = x, log = FALSE, do.its = FALSE)

plot(pdf)
```

---

pit *Probability Integral Transform.*

Description

Method returning the predictive probability integral transform (PIT) in-sample or of a vector of points consider as one step ahead draws (t = T + 1).

Usage

```
pit(object, x, theta, y, do.norm = FALSE, do.its = FALSE)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.
x	Vector (of size N) of pointsevaluated at t = T + 1 (used when do.its = FALSE).
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
y	Vector (of size T) of observations (not required when using a fit object).
do.norm	Boolean indicating if the PIT values are transformed into standard Normal variate. (Default: do.norm = FALSE).
do.its	Boolean indicating if the in-sample pit is returned. (Default: do.its = FALSE)

Details

If a matrix of MCMC posterior draws estimates is given, the Bayesian probability integral transform is calculated. If do.its = FALSE, the points x are evaluated as t = T + 1 realizations and the method uses the variance estimate at t = T + 1. If do.its = TRUE, y is evaluated using their respective variance estimate at each time t. The do.norm argument transforms the PIT value into Normal variates so that normality test can be done.

Value

A list of class MSGARCH_PIT containing three components:

- pit:
  If do.its = FALSE: probability integral transform of the points x at t = T + 1 or Normal variate derived from the probability integral transform of x (vector of size N).
  If do.its = TRUE: In-sample probability integral transform or Normal variate derived from the probability integral transform of y (vector of size T or matrix of size M x T).
- x:
  If do.its = FALSE: Vector (of size N) of at points evaluated at t = T + 1.
  If do.its = TRUE: Vector (of size T) of observations.
- do.its: Orinigal user inputed do.its for reference.

The class MSGARCH_PIT contains the plot method only if do.its = FALSE.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# run pit method in-sample
```

```
pit.its = MSGARCH::pit(object = fit, do.norm = FALSE, do.its = TRUE)

plot(pit.its)

# generate random draws at T + 1 from model
set.seed(123)
sim.ahead = MSGARCH::simahead(object = fit, n = 1, m = 100)

x = sim.ahead$draws

# run pit method on random draws at T + 1 from model
pit = MSGARCH::pit(object = fit, x = x, do.norm = FALSE)

plot(pit)
```

`pred` *Predictive function.*

Description

Method returning the predictive probability density in-sample or of a vector of points consider as one step ahead draws (`t = T + 1`).

Usage

```
pred(object, x, theta, y, log = FALSE, do.its = FALSE)
```

Arguments

`object`	Model specification of class `MSGARCH_SPEC` created with `create.spec` or fit object of type `MSGARCH_MLE_FIT` created with `fit.mle` or `MSGARCH_BAY_FIT` created with `fit.bayes`.
`x`	Vector (of size N) of points evaluated at `t = T + 1` (used when `do.its = FALSE`).
`theta`	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
`y`	Vector (of size T) of observations (not required when using a fit object).
`log`	Boolean indicating if the log-density is returned. (Default: `log = FALSE`)
`do.its`	Boolean indicating if the in-sample predictive is returned. (Default: `do.its = FALSE`)

Details

If a matrix of MCMC posterior draws estimates is given, the Bayesian Probability integral transform is calculated. If `do.its = FALSE`, the points `x` are evaluated as `t = T + 1` realizations and the method uses the variance estimate at `t = T + 1`. If `do.its = TRUE`, `y` is evaluated using their respective variance estimate at each time `t`.

Value

A list of class MSGARCH_PRED containing three components:

- pred:
  If do.its = FALSE: (Log-)Predictive of of the points x at t = T + 1 (vector of size N).
  If do.its = TRUE: In-sample Predictive of y (vector of size T or matrix of size M x T).
- x:
  If do.its = FALSE: Vector (of size N) of points evaluated at t = T + 1.
  If do.its = TRUE: Vector (of size T) of observations.
- do.its: Orinigal user inputed do.its for reference.

The class MSGARCH_PRED contains the plot method only if do.its = FALSE.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation using DEoptim intialization
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# run pred method in-sample
pred.its = MSGARCH::pred(object = fit, log = TRUE, do.its = TRUE)

sum(pred.its$pred, na.rm = TRUE)

# create mesh
x = seq(-3,3,0.01)

# run pred method on mesh at T + 1
pred = MSGARCH::pred(object = fit, x = x, log = TRUE, do.its = FALSE)

plot(pred)
```

Pstate *Filtered state probabilities.*

Description

Method returning the filtered probabilities of the states.

Usage

```
Pstate(object, theta, y)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
y	Vector (of size T) of observations (not required when using a fit object).

Details

If a matrix of parameter estimates is given, each parameter estimate (each row) is evaluated individually.

Value

Filtered state probabilities of class MSGARCH_PSTATE (array of size (T + 1) x M x K). The class MSGARCH_PSTATE contains the plot method.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# compute the filtered state probabilities
Pstate  = MSGARCH::Pstate(object = fit)

plot(Pstate)
```

---

risk *Value-at-Risk And Expected-shortfall.*

Description

Method returning the Value-at-Risk and Expected-shortfall in-sample or at t = T + 1 based on the predictive density.

Usage

```
risk(object, theta, y, level = c(0.95, 0.99), ES = TRUE, do.its = FALSE,
  ctr = list(n.mesh = 500, tol = 1e-04, itermax = 5))
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
y	Vector (of size T) of observations (not required when using a fit object).
level	Vector (of size R) of Value-at-risk and Expected-shortfall levels. (Default: level = c(0.95,0.99))
ES	Boolean indicating if Expected-shortfall is also calculated. (Default: ES = TRUE)
do.its	Boolean indicating if the in-sample risk estimators are returned. (Default: do.its = FALSE)
ctr	List of control parameters for risk evaluation.

Details

If a matrix of MCMC posterior draws estimates is given, the Bayesian Value-at-Risk and Expected-shortfall are calculated. If do.its = FALSE, x the risk estimator at t = T + 1, the method uses the variance estimated at t = T + 1. If do.its = TRUE, the in-sample risk estimator are calculated.

Value

A list of class MSGARCH_RISK containing two or three components:

- VaR :
  If do.its = FALSE: Value-at-Risk at t = T + 1 at the chosen levels (vector of size R).
  If do.its = TRUE: In-sample Value-at-Risk at the chosen levels (Matrix of size T x R).
- ES :
  If do.its = FALSE: Expected-shortfall at t = T + 1 at the chosen levels (vector of size R).
  If do.its = TRUE: In-sample Expected-shortfall at the chosen levels (Matrix of size T x R).
- y : Vector (of size T) of observations.

The MSGARCH_RISK contains the plot method. The Bayesian risk estimator can take long time to calculate depending on the size of the MCMC chain.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# compute the Value-at-Risk and Expected-shortfall
# Risk estimation in-sample
risk.its = MSGARCH::risk(object = fit, level = 0.95, ES = FALSE, do.its = TRUE)
```

```
plot(risk.its)

# Risk estimation at T + 1
risk = MSGARCH::risk(object = fit, level = 0.95, ES = FALSE, do.its = FALSE)
```

`sim` *Process simulation method.*

Description

Method simulating a MSGARCH process.

Usage

```
sim(object, n, m, theta, burnin = 500)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec.
n	Simulation length. (Default: n = 1000)
m	Number of simulations. (Default: m = 1)
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
burnin	Burnin period discarded (first simulation draws). (Default: burnin = 500)

Details

If a matrix of parameters estimates is given, each parameter estimates is evaluated individually and m = M. The difference between sim and simahead is that sim starts the simulation a t = 0 creating an entire new process while simahead starts the simulation at t = T + 1 taking in consideration all the information available in the original time series y.

Value

A list of class MSGARCH_SIM containing two components.

- draws: Matrix (of size M x n) of simulated draws.
- state: Matrix (of size M x n) of simulated states.

The MSGARCH_SIM class contains the plot method.

Examples

```
## Not run:
require("MSGARCH")
# create model specification
spec = MSGARCH::create.spec()

# generate process
set.seed(123)
```

```
sim = MSGARCH::sim(object = spec, n = 1000, m = 1, theta = spec$theta0, burnin = 500)

plot(sim)

## End(Not run)
```

---

simahead *Step ahead simulation method.*

Description

Method returning step ahead simulation up to time n.

Usage

```
simahead(object, n, m, theta, y)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.
n	Number of steps ahead time steps. (Default: n = 1)
m	Number of simulations. (Default: m = 1)
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
y	Vector (of size T) of observations (not required when using a fit object).

Details

If a matrix of parameters estimates is given, each parameter estimates is evaluated individually and m = M. The MSGARCH_SIM class contains the plot method. The difference between sim and simahead is that sim starts the simulation a t = 0 creating an entire new process while simahead starts the simulation at t = T + 1 taking in consideration all the information available in the original time series y.

Value

A list of class MSGARCH_SIM containing two components:

- draws: Matrix (of size m x n) of step ahead simulated draws.
- state: Matrix (of size m x n) of step ahead simulated states.

The MSGARCH_SIM class contains the plot method.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# generate random draws
set.seed(123)
simahead = MSGARCH::simahead(object = fit, n = 30, m = 100)

plot(simahead)
```

---

sp500 *Log return of the S&P 500 index closing Value*

Description

The S&P 500 index closing value log return from 1998-01-01 to 2015-12-31 from Yahoo Finance https://finance.yahoo.com/.

Usage

```
data("sp500")
```

Format

Matrix containing 4,529 observations.

Source

Yahoo Finance https://finance.yahoo.com/

---

transmat *Transition Matrix.*

Description

Method returning the transition matrix.

Usage

```
transmat(object, theta, n)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle.
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.
n	Number of steps ahead. (Default: n = 1

Value

A matrix (of size K x K) in the case of a Markov-Switching model or a vector (of size K) in the case of a Mixture model. The columns indicates the starting states while the rows indicates the transition states.

Examples

```
require("MSGARCH")
# load data
data("sp500")
sp500 = sp500[1:1000]

# create model specification
spec = MSGARCH::create.spec()

# fit the model on the data with ML estimation
set.seed(123)
fit = MSGARCH::fit.mle(spec = spec, y = sp500, ctr = list(do.init = FALSE))

# Extract the transition matrix 10 steps ahead
transmat.mle = MSGARCH::transmat(fit, n = 10)

print(transmat.mle)
```

---

unc.vol *Unconditional volatility of each regime.*

Description

Method returning the unconditional volatility of the process in each state.

Usage

```
unc.vol(object, theta)
```

Arguments

object	Model specification of class MSGARCH_SPEC created with create.spec or fit object of type MSGARCH_MLE_FIT created with fit.mle or MSGARCH_BAY_FIT created with fit.bayes.
theta	Vector (of size d) or matrix (of size M x d) of parameter estimates (not required when using a fit object) where d must have the same length as the default parameters of the specification.

Details

If a matrix of parameter estimates (each row) is given, each parameter estimates is evaluated individually.

Value

Unconditional volatility (vector of size K or matrix of size M x K) of each regime.

Examples

```
require("MSGARCH")
# create model specification
spec = MSGARCH::create.spec()

# compute the unconditional volatility in each regime
unc.vol = MSGARCH::unc.vol(object = spec, theta = spec$theta0)
```

Index