Package ‘MSGARCH’

May 16, 2018

Type Package
Title Markov-Switching GARCH Models
Version 2.3
Date 2018-05-15
Author David Ardia [aut],
      Keven Bluteau [aut, cre],
      Kris Boudt [ctb],
      Leopoldo Catania [aut],
      Alexios Ghalanos [ctb],
      Brian Peterson [ctb],
      Denis-Alexandre Trottier [aut]
Maintainer Keven Bluteau <Keven.Bluteau@unine.ch>
Description Fit (by Maximum Likelihood or MCMC/Bayesian), simulate, and forecast vari-
      ous Markov-Switching GARCH models as described in Ar-
License GPL (>= 2)
BugReports https://github.com/keblu/MSGARCH/issues
URL https://github.com/keblu/MSGARCH
Imports Rcpp, coda, methods, zoo, expm, fanplot, MASS, numDeriv
LinkingTo Rcpp, RcppArmadillo
Suggests mcmc, testthat
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-05-16 05:07:38 UTC

R topics documented:

  MSGARCH-package .................................................. 2
  CreateSpec ......................................................... 3
Description

The R package MSGARCH implements a comprehensive set of functionalities for Markov-switching GARCH (Haas et al. 2004a) and Mixture of GARCH (Haas et al. 2004b) models. This includes fitting, filtering, forecasting, and simulating. Other functions related to Value-at-Risk and Expected-Shortfall are also available.

The main functions of the package are coded in C++ using Rcpp (Eddelbuettel and Francois, 2011) and RcppArmadillo (Eddelbuettel and Sanderson, 2014).

MSGARCH focuses on the conditional variance (and higher moments) process. Hence, there is no equation for the mean. Therefore, you must pre-filter via AR(1) before applying the model.

The MSGARCH package implements a variety of GARCH specifications together with several 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)). When optimization is performed, we ensure that the variance in each regime is covariance-stationary and strictly positive (refer to the vignette for more information).


The authors acknowledge Google for financial support via the Google Summer of Code 2016 & 2017, the International Institute of Forecasters and Industrielle-Alliance.

Author(s)

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

• David Ardia <david.ardia.ch@gmail.com>
• Leopoldo Catania <leopoldo.catania@econ.au.dk>
CreateSpec

- Denis-Alexandre Trottier <denis-alexandre.trottier.1@ulaval.ca>

Other contributors:
- Kris Boudt <kris.boudt@vub.ac.be> [contributor]
- Alexios Ghalanos <aleXios@4dscape.com> [contributor]
- Brian Peterson <brian@braverock.com> [contributor]

References


See Also

Useful links:
- https://github.com/keblu/MSGARCH
- Report bugs at https://github.com/keblu/MSGARCH/issues

createSpec

Model specification.

Description

Creates a model specification before fitting and using the MSGARCH functionalities.

Usage

createSpec(variance.spec = list(model = c("sGARCH", "sGARCH")),
distribution.spec = list(distribution = c("norm", "norm")),
switch.spec = list(do.mix = FALSE, K = NULL), constraint.spec = list(fixed = list(), regime.const = NULL), prior = list(mean = list(), sd = list()))
Arguments

**variance.spec** list with element `model`. `model` is a character vector (of size K, number of regimes) with the variance model specifications. Valid models are "sARCH", "sGARCH", "eGARCH", "gjrGARCH", and "tGARCH" (see *Details*). Default: `model = c("sGARCH", "sGARCH")`.

**distribution.spec** list with element `distribution`. `distribution` is a character vector (of size K) of conditional distributions. Valid distributions are "norm", "snorm", "std", "sstd", "ged", and "sged" (see *Details*). The vector must be of the same length as the models' vector in `variance.spec`. Default: `distribution = c("norm", "norm")`.

**switch.spec** list with element `do.mix` and `k`. `do.mix` is a logical indicating if the specification is a mixture type. If `do.mix = TRUE`, a Mixture of GARCH is created, while if `do.mix = FALSE`, a Markov-Switching GARCH is created (see *Details*). Default: `do.mix = FALSE`. `k` is a optional numeric scalar indicating the number of regime. In the case where a single regime is specified in `variance.spec` and `distribution.spec`, this parameter allows to automatically expand this single regime to K similar regimes without the need to explicitly define them in `variance.spec` and `distribution.spec` (see *Examples*).

**constraint.spec** list with element `fixed` and `regime.const`. Only one of `fixed` and `regime.const` can be set by the user as it is not allowed to set both at the same time. `fixed` is a list with numeric entries and named elements. This argument controls for fixed parameters defined by the user. The names of the entries in the list have to coincide with the names of the model parameters. For instance, if `constraint.spec = list(fixed = list(beta_1 = 0))`, `beta_1` will be fixed to 0 during optimization. `regime.const` is a character vector. This argument controls for the parameters which are set equal across regimes. The names of the entries in the list have to coincide with the names of the model parameters minus the regime indicator. For instance, if `constraint.spec = list(regime.const = c("beta"))`, all the parameters named `beta` will be the same in all regimes during optimization.

**prior** list with element `mean` and `sd`. The element `mean` and `sd` are list with numeric and named elements that allow to adjust the prior mean and standard deviation of the truncated Normal prior. The names of the entries in the lists have to coincide with the names of the model parameters. For instance, if `prior = list(mean = list(beta_1 = 0.7), sd = list(beta_1 = 0.1))`, the prior mean of `beta_1` will be set to 0.7 while the prior standard deviation will set to 0.1.

Details

The Markov-Switching specification is based on the Haas et al. (2004a) MSGARCH specification. It is a MSGARCH model that is separated in K single-regime specifications which are updated in
parallel. Under the Haas et al. (2004a) specification, the conditional variance is a function of past data and the current state. The Mixture of GARCH option (do.mix = TRUE) 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.

For the models, "sARCH" is the ARCH(1) model (Engle, 1982), "sGARCH" the GARCH(1,1) model (Bollerslev, 1986), "eGARCH" the EGARCH(1,1) model (Nelson, 1991), "gjrGARCH" the GJR(1,1) model (Glosten et al., 1993), and "tGARCH" the TGARCH(1,1) model (Zakoian, 1994).

For the distributions, "norm" is the Normal distribution, "std" the Student-t distribution, and "ged" the GED distribution. Their skewed version, implemented via the Fernandez and Steel (1998) transformation, are "snorm", "sstd" and "sged". Please see Ardia et al. (2017) for more details on the models and distributions.

The user must choose between fixed or regime.const in contraint.spec as both cannot be set at the same time. The list fixed.pars will ensure that the chosen fixed parameters will be fixed during optimization according to the values set by the user. Thus only the non-fixed parameters are optimized. The vector regime.const will ensure that the chosen parameters will be the same across regime during optimization.

The list mean and sd in prior will adjust the prior mean and prior standard deviation of the truncated Normal prior for MCMC estimation via FitMCMC according to the inputed prior mean and standard deviation. Those prior means and standard deviations that are not set will take on preset default values (a mean of zero and a variance of 1,000).

### Value

A list of class MSGARCH_SPEC with the following elements:

- `par0`: Vector (of size d) of default parameters.
- `is.mix`: Logical indicating if the specification is a mixture.
- `K`: Number of regimes.
- `lower`: Vector (of size d) of lower parameters’ bounds.
- `upper`: Vector (of size d) of upper parameters’ 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.
- `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.
- `fixed.pars`: List of user inputed fixed parameters.
- `regime.const.pars`: Vector of user inputed parameter set equal across regimes.
- `regime.fixed.pars`: Logical indicating if there is any fixed parameter set by the user.
- `regime.const.pars.bool`: Logical indicating if there is any parameters equal across regime set by the user.
The `MSGARCH_SPEC` class has the following methods:

- **simulate**: Simulation.
- **Volatility**: In-sample conditional volatility.
- **predict**: Forecast of the conditional volatility (and predictive distribution).
- **UncVol**: Unconditional volatility.
- **PredPdf**: Predictive density (pdf).
- **PIT**: Probability Integral Transform.
- **Risk**: Value-at-Risk and Expected-Shortfall.
- **State**: State probabilities (smoothed, filtered, predictive, Viterbi).
- **FitML**: Maximum Likelihood estimation.
- **FitMCMC**: Bayesian estimation.
- **print** and **summary**: Summary of the created specification.

**References**


**Examples**

```r
# create a Markov-switching specification
# MS-GARCH(1,1)-GJR(1,1)-Student
spec <- CreateSpec(variance.spec = list(model = c("sGARCH","gjrGARCH")),
                   distribution.spec = list(distribution = c("std","std")),
                   switch.spec = list(do.mix = FALSE))

print(spec)
```
# create a 3-regime Markov-switching specification with the help of variable K
# MS(3)-GARCH(1,1)- Student
spec <- CreateSpec(variance.spec = list(model = c("sGARCH")),
distribution.spec = list(distribution = c("std")),
switch.spec = list(do.mix = FALSE, K = 3))
print(spec)

# create a mixture specification
# MIX-GARCH(1,1)-GJR(1,1)-Student
spec <- CreateSpec(variance.spec = list(model = c("sGARCH","gjrGARCH")),
distribution.spec = list(distribution = c("std","std")),
switch.spec = list(do.mix = TRUE))
print(spec)

# setting fixed parameter for the sGARCH beta parameter
# MS-GARCH(1,1)-GJR(1,1)-Student with beta_1 fixed to 0
spec <- CreateSpec(variance.spec = list(model = c("sGARCH","gjrGARCH")),
distribution.spec = list(distribution = c("std","std")),
switch.spec = list(do.mix = FALSE),
constraint.spec = list(fixed = list(beta_1 = 0)))
print(spec)

# setting restriction for the shape parameter of the Student-t across regimes
# MS-GARCH(1,1)-GJR(1,1)-Student with shape parameter constraint across regime
spec <- CreateSpec(variance.spec = list(model = c("sGARCH","gjrGARCH")),
distribution.spec = list(distribution = c("std","std")),
switch.spec = list(do.mix = FALSE),
constraint.spec = list(regime.const = c("nu")))
print(spec)

# setting custom parameter priors for the beta parameters
# MS-GARCH(1,1)-GJR(1,1)-Student with prior modification
spec <- CreateSpec(variance.spec = list(model = c("sGARCH","gjrGARCH")),
distribution.spec = list(distribution = c("std","std")),
switch.spec = list(do.mix = FALSE),
prior = list(mean = list(beta_1 = 0.9, beta_2 = 0.3),
            sd = list(beta_1 = 0.05, beta_2 = 0.01)))
print(spec)

dem2gbp

**DEM/GBP exchange rate log-returns**

### Description

The vector `dem2gbp` contains daily observations of the Deutschmark vs British Pound foreign exchange rate log-returns. This dataset has been promoted as an informal benchmark for GARCH time-series software validation. See McCullough and Renfro (1999), and Brooks, Burke, and Persand (2001) for details. The nominal returns are expressed in percent as in Bollerslev and Ghysels (1996). The sample period is from January 3, 1984, to December 31, 1991, for a total of 1974 observations.
Usage

data("dem2gbp")

Format

vector of size 1,974.

References


DIC Deviance Information Criterion (DIC).

Description

Method which computes the Deviance Information Criterion (DIC) from a fit object of type *msgarch_mcmc_fit* created with *fitMCMC*.

Usage

DIC(fit)

## S3 method for class 'MSGARCH_MCMC_FIT'
DIC(fit)

Arguments

fit Fit object of type *MSGARCH_MCMC_FIT* created with *fitMCMC*.

Details

Computes the Deviance information criterion of Spiegelhalter et al. (2002).

Value

A list with the following elements:

- 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


Examples

```r
# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on data by MCMC
set.seed(123)
fit <- fitMCMC(spec = spec, data = SMI, ctr = list(nburn = 500L, nmcmc = 500L))

# compute DIC
DIC(fit)
```

**ExtractStateFit**  
*Single-regime model extractor.*

**Description**

Extracts each regime from a fitted multiple regime specification and creates a fitted object for each extracted regime.

**Usage**

```r
ExtractStateFit(object)
```

### S3 method for class 'MSGARCH_ML_FIT'

```r
ExtractStateFit(object)
```

### S3 method for class 'MSGARCH_MCMC_FIT'

```r
ExtractStateFit(object)
```

**Arguments**

- `object`  
  Fit object of type MSGARCH_ML_FIT created with `FitML` or MSGARCH_MCMC_FIT created with `FitMCMC`.

**Value**

A list of K elements where each element is a fit object of type MSGARCH_ML_FIT or MSGARCH_MCMC_FIT.
Examples

```r
# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on the data with ML estimation
fit <- FitML(spec = spec, data = SMI)
SR.fit <- ExtractStateFit(fit)
print(SR.fit)
```

---

**FitMCMC**

*MCMC/Bayesian estimation.*

---

**Description**

Method that performs MCMC/Bayesian estimation of a `msgarch_spec` object on a set of observations.

**Usage**

```r
FitMCMC(spec, data, ctr = list())
```

**Arguments**

- **spec**
  Model specification of class `MSGARCH_SPEC` created with `CreateSpec`.

- **data**
  Vector (of size T) of observations.

- **ctr**
  A list of control parameters:
  - `par0`: Vector (of size d) where d must have the same length as the default parameters of the specification. It is the starting value for the chain (if empty the the method automatically set starting parameters; see *Details*).
  - `nburn` (integer >= 0): Number of discarded draws. (Default: `nburn = 5000L`)
  - `nmc` (integer > 0): Number of draws. (Default: `nmc = 10000L`)
  - `nthin` (integer > 0): Thinning factor (every nthin draws are kept). (Default: `nthin = 10L`)
  - `do.sort` (bool): Logical indicating if the MCMC draws are post-sorted following Geweke (2007). By default, `do.sort = TRUE`, such that the MCMC draws are ordered to ensure that unconditional variance is an increasing function of the regime (identification constraint). If the user sets `do.sort = FALSE`, no sorting is imposed, and label switching can occur (see *Details*).
  - `SamplerFUN`: Custom MCMC sampler (see *Details*).  

---
Details

The total number of draws is equal to $n_{mcmc} / nthin$. The MCMC/Bayesian estimation relies on an Rcpp implementation of the adaptive sampler of Vihola (2012). The implementation is based on the R package adaptMCMC (Andreas, 2012). Starting values when $par\theta$ is not provided are chosen automatically before sampling (see Ardia et al. (2017) for more details).

SampelrFUN allows for a custom sampler to be used. The function must take the form:

```r
function(f_posterior, data, spec, par\theta, ctr).
```

where $f_{\text{posterior}}$ is the function to optimize, $data$ is the data, $spec$ is the specification, $par\theta$ are the starting parameters, and $ctr$ are the control parameters. The inputs $spec$ and $data$, must be passed as inputs in the sampler (see *Examples*). The custom sampler must output a matrix containing the MCMC chain.

When $do.sort = \text{TRUE}$, sorting of each MCMC draw conditional on the unconditional variance is done across homogeneous regime specification.

Value

A list of class MSGARCH_MCMC_FIT with the following elements:

- **par**: The MCMC chain (matrix from the R package coda (Plummer et al., 2006) of size $n_{mcmc} / nthin \times d$).
- **accept**: Acceptance rate of the sampler.
- **spec**: Model specification of class MSGARCH_SPEC created with CreateSpec.
- **data**: Vector (of size $T$) of observations.
- **ctr**: list of the control used for the fit.

The MSGARCH_MCMC_FIT with the following methods:

- **DIC**: Deviance Information Criterion (DIC).
- **simulate**: Simulation.
- **Volatility**: In-sample conditional volatility.
- **predict**: Forecast of the conditional volatility (and predictive distribution).
- **UncVol**: Unconditional volatility.
- **PredPdf**: Predictive density (pdf).
- **PIT**: Probability Integral Transform.
- **Risk**: Value-at-Risk and Expected-Shortfall.
- **State**: State probabilities (smoothed, filtered, predictive, Viterbi).
- **ExtractStateFit**: Single-regime model extractor.
- **summary**: Summary of the fit.
References


Examples

# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on the data by MCMC
set.seed(123)
fit <- FitMCMC(spec = spec, data = SMI, ctr = list(nburn = 500L, nmcmc = 500L, nthin = 1L))
summary(fit)

# custom sampler example
## Not run:
library("mcmc")
f_MCMC <- function(f_posterior, data, spec, par0, ctr){
  par <- mcmc::metrop(f_posterior, initial = par0, nbatch = ctr$nmcmc + ctr$nburn,
                      data = data, spec = spec)$batch
  colnames(par) = names(par0)
  return(par)
}
set.seed(123)
fit <- FitMCMC(spec, data = SMI, ctr = list(SamplerFUN = f_MCMC,
                                           nburn = 500L, nmcmc = 500L, nthin = 1L))
summary(fit)
## End(Not run)
**Description**

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

**Usage**

```r
FitML(spec, data, ctr = list())
```

**Arguments**

- `spec` Model specification created with `CreateSpec`.
- `data` Vector (of size T) of observations.
- `ctr` A list of control parameters:
  - `par0`: Vector (of size d) where d must have the same length as the default parameters of the specification. It is the starting value for the optimizer (if empty the the method automatically set starting parameters; see *Details*).
  - `do.se` Logical. Should standard errors be computed? (Default: `do.se = TRUE`).
  - `do.plm` Logical. If `do.plm = FALSE`, parameter transformation during the optimization step is performed without ensuring stationarity for the volatility processes. For combinations of parameters that do not imply stationarity the likelihood value is fixed at -1e10. If fixed is defined in the list `contraint.spec` of `CreateSpec`, `do.plm = TRUE` is used. (Default: `do.plm = FALSE`)
  - `OptimFUN`: Custom optimization function (see *Details*).

**Details**

By default, `OptimFUN` is set such that optimization is done via the well-known Broyden- Fletcher-Goldfarb-Shanno (BFGS) algorithm using the `optim` function with `method = "BFGS"`. Starting values when `par0` is not provided are chosen automatically before optimization (see Ardia et al. (2017) for more details)

`OptimFUN` allows for a custom optimizer to be used. The function must take the form:

```r
function(vPw, f_nll, spec, data, do.plm)
```

where `vPw` are starting parameters (transformed), `f_nll` is the function to be minimize, `spec` is the specification, `data` is the data, and `do.plm` the originally inputed or default `do.plm`. The inputs `spec`, `data`, and `do.plm` must be passed as inputs in the optimizer (see *Examples*). It must output a list with the following elements:

- `value`: Optimal negative log-likelihood.
- `par`: Optimal parameters.
Value

A list of class MSGARCH_ML_FIT with the following elements:

- `par`: Vector (of size d) of optimal parameters.
- `loglik`: Log-likelihood of y given the optimal parameters.
- `inference`: List with elements `MatCoef` and `Hessian`. `MatCoef` is a matrix (of size d x 4) with optimal parameter estimates, standard errors, t-stats, and p-values. `Hessian` is the Hessian (matrix of size d x d) of the negative log-likelihood function evaluated at the optimal parameter estimates `par`.
- `spec`: Model specification of class MSGARCH_SPEC created with `CreateSpec`.
- `data`: Vector (of size T) of observations.
- `ctr`: List of the control used for the fit.

The MSGARCH_ML_FIT with the following methods:

- `AIC`: Akaike Information Criterion (AIC).
- `BIC`: Bayesian Information Criterion (BIC).
- `simulate`: Simulation.
- `Volatility`: In-sample conditional volatility.
- `predict`: Forecast of the conditional volatility (and predictive distribution).
- `UncVol`: Unconditional volatility.
- `PredPdf`: Predictive density (pdf).
- `PIT`: Probability Integral Transform.
- `Risk`: Value-at-Risk and Expected-Shortfall.
- `State`: State probabilities (smoothed, filtered, predictive, Viterbi).
- `ExtractStateFit`: Single-regime model extractor.
- `summary`: Summary of the fit.

Examples

```r
# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on the data by ML
fit <- FitML(spec = spec, data = SMI)
summary(fit)

# custom optimizer example
## Not run:
f_custom_optim <- function(vPw, f_nll, spec, data, do.plm){
  out <- stats::optim(vPw, f_nll, spec = spec, data = data,
                     do.plm = do.plm, method = "Nelder-Mead")
  return(out)
```
PIT

Probability integral transform.

Description

Method returning the probability integral transform (PIT).

Usage

PIT(object, ...)

## S3 method for class 'MSGARCH_SPEC'
PIT(object, x = NULL, par = NULL, data = NULL,
    do.norm = FALSE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,
    ctr = list(), ...)

## S3 method for class 'MSGARCH_ML_FIT'
PIT(object, x = NULL, newdata = NULL,
    do.norm = TRUE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,
    ctr = list(), ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
PIT(object, x = NULL, newdata = NULL,
    do.norm = TRUE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,
    ctr = list(), ...)

Arguments

object Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.

... Not used. Other arguments to PIT.

x Vector (of size n). Used when do.its = FALSE.

par Vector (of size d) or matrix (of size nmc x d) of parameter estimates where d must have the same length as the default parameters of the specification.

data Vector (of size T) of observations.

do.norm Logical indicating if the PIT values are transformed into standard Normal variate. (Default: do.norm = FALSE)
do.its Logical indicating if the in-sample PIT is returned. (Default: do.its = FALSE)

nahead Scalar indicating the number of step-ahead evaluation. Valid only when do.its = FALSE. (Default: nahead = 1L)

do.cumulative Logical indicating if the PIT is computed on the cumulative simulations (typically log-returns, as they can be aggregated). Only available for do.its = FALSE. (Default: do.cumulative = FALSE)

ctr A list of control parameters:
- nsim (integer >= 0): Number indicating the number of simulation done for the evaluation of the PIT at nahead > 1. (Default: nsim = 10000L)

ewdata Vector (of size T*) of new observations. (Default: newdata = NULL)

Details

If a matrix of MCMC posterior draws is given, the Bayesian probability integral transform is calculated. Two or more step-ahead probability integral transform are estimated via simulation of nsim paths up to t = T + T* + nahead. The empirical probability integral transforms is then inferred from these simulations.

If do.its = FALSE, the vector x are evaluated as t = T + T* + 1, ... , t = T + T* + nahead realizations.

If do.its = TRUE, x is evaluated at each time t up to time t = T + T*.

Finally if x = NULL the vector data is evaluated for sample evaluation of the PIT. The do.norm argument transforms the PIT value into Normal variates so that normality test can be done.

Value

A vector or matrix of class MSGARCH_PIT.

If do.its = FALSE: Probability integral transform of the points x at t = T + T* + 1, ... , t = T + T* + nahead or Normal variate derived from the probability integral transform of x (matrix of size nahead x n).

If do.its = TRUE: In-sample probability integral transform or Normal variate derived from the probability integral transform of data if x = NULL (vector of size T + T*) or in-sample probability integral transform or Normal variate derived from the probability integral transform of x (matrix of size (T + T*) x n).

Examples

```r
# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on the data by ML
fit <- FitML(spec = spec, data = SMI)

# run PIT method in-sample
pit.its <- PIT(object = fit, do.norm = TRUE, do.its = TRUE)
```
predict.MSGARCH_SPEC

# diagnostic of PIT with qqnorm
qqnorm(pit.its)
qqline(pit.its)

# simulate a serie from the model
set.seed(123)
sim.series <- simulate(object = spec, par = fit$par, nahead = 1000L, nsim = 1L)
sim.series <- as.vector(sim.series$draw)

# run PIT method on the simulated serie with the true par
pit.x <- PIT(object = spec, par = fit$par, data = sim.series, do.norm = TRUE, do.its = TRUE)
qqnorm(pit.x)
qqline(pit.x)

Description

Conditional volatility (and predictive distribution) forecasts.

Usage

## S3 method for class 'MSGARCH_SPEC'
predict(object, newdata = NULL, nahead = 1L,
do.return.draw = FALSE, par = NULL, do.cumulative = FALSE,
ctr = list(), ...)

## S3 method for class 'MSGARCH_ML_FIT'
predict(object, newdata = NULL, nahead = 1L,
do.return.draw = FALSE, do.cumulative = FALSE, ctr = list(), ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
predict(object, newdata = NULL, nahead = 1L,
do.return.draw = FALSE, do.cumulative = FALSE, ctr = list(), ...)

Arguments

object  Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.
newdata Vector (of size T*) of new observations. (Default newdata = NULL)
nahead Scalar indicating the number of step-ahead evaluation.
do.return.draw Are simulation draws from the predictive distribution returned? (Default do.return.draw = FALSE)
par Vector (of size d) or matrix (of size nmc x d) of parameter estimates where d must have the same length as the default parameters of the specification.
do.cumulative  Logical indicating if the conditional volatility prediction is computed on the cumulative simulations (typically log-returns, as they can be aggregated). (Default: do.cumulative = FALSE)

ctr        A list of control parameters:
               • nsim (integer >= 0): Number indicating the number of simulation done for the conditional volatility forecast at nahead > 1. (Default: nsim = 10000)

Details

If a matrix of MCMC posterior draws is given, the Bayesian predictive conditional volatility (and predictive distribution) forecasts are returned.

Value

A list of class MSGARCH_FORECAST with the following elements:

• vol: Conditional volatility forecast (vector of size nahead).
• draw: If do.return.draw = TRUE:
  Draws sampled from the predictive distributions (matrix of size nahead x nsim).
  If do.return.draw = FALSE:
    NULL

The MSGARCH_FORECAST class contains the plot method.

Examples

# create specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# predict from specification
par <- c(0.1, 0.1, 0.8, 0.2, 0.1, 0.8, 0.99, 0.01)
set.seed(1234)
pred <- predict(object = spec, par = par, newdata = SMI, nahead = 5L)
head(pred)
plot(pred)

# predict from ML fit
fit <- FitML(spec = spec, data = SMI)
set.seed(1234)
pred <- predict(object = fit, nahead = 5L, do.return.draw = TRUE)
head(pred)
plot(pred)

## Not run:
set.seed(1234)
fit <- FitMCMC(spec = spec, data = SMI)
pred <- predict(object = fit, nahead = 5L, do.return.draw = TRUE)
predpdf

Predictive density.

Description

Method returning the predictive density (pdf).

Usage

PredPdf(object, ...)  

## S3 method for class 'MSGARCH_SPEC'
PredPdf(object, x = NULL, par = NULL, data = NULL,  
log = FALSE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,  
ctr = list(), ...)

## S3 method for class 'MSGARCH_ML_FIT'
PredPdf(object, x = NULL, newdata = NULL,  
log = FALSE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,  
ctr = list(), ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
PredPdf(object, x = NULL, newdata = NULL,  
log = FALSE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,  
ctr = list(), ...)

Arguments

object               Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.
...                   Not used. Other arguments to PredPdf.
x                   Vector (of size n). Used when do.its = FALSE.
par                  Vector (of size d) or matrix (of size nmcmc x d) of parameter estimates where d must have the same length as the default parameters of the specification.
data                 Vector (of size T) of observations.
log                   Logical indicating if the log-density is returned. (Default: log = FALSE)
do.its               Logical indicating if the in-sample predictive is returned. (Default: do.its = FALSE)
nahead               Scalar indicating the number of step-ahead evaluation. Valid only when do.its = FALSE. (Default: nahead = 1L)
do.cumulative Logical indicating if predictive density is computed on the cumulative simulations (typically log-returns, as they can be aggregated). Only available for do.its = FALSE. (Default: do.cumulative = FALSE)

ctr A list of control parameters:

- nsim (integer >= 0): Number indicating the number of simulation done for the evaluation of the density at nahead > 1. (Default: nsim = 10000L)

newdata Vector (of size T*) of new observations. (Default newdata = NULL)

Details

If a matrix of MCMC posterior draws is given, the Bayesian predictive probability density is calculated. Two or more step-ahead predictive probability density are estimated via simulation of nsim paths up to \( t = T + T* + \text{nahead} \). The predictive distribution are then inferred from these simulations via a Gaussian Kernel density. If do.its = FALSE, the vector \( x \) are evaluated as \( t = T + T* + 1, \ldots, t = T + T* + \text{nahead} \) realization.

If do.its = TRUE and \( x \) is evaluated at each time \( t \) up top time \( t = T + T* \).

Finally, if \( x = \text{NULL} \) the vector \( \text{data} \) is evaluated for sample evaluation of the predictive density ((log-)likelihood of each sample points).

Value

A vector or matrix of class MSGARCH_PRED.

If do.its = FALSE: (Log-)predictive of the points \( x \) at \( t = T + T* + 1, \ldots, t = T + T* + \text{nahead} \) (matrix of size nahead x n).

If do.its = TRUE: In-sample predictive of \( \text{data} \) if \( x = \text{NULL} \) (vector of size \( T + T* \)) or in-sample predictive of \( x \) (matrix of size \( (T + T* \) x n).

Examples

```r
# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on the data by ML
fit <- FitML(spec = spec, data = SMI)

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

# create a mesh
x <- seq(-3,3,0.01)

# run PredPdf method on mesh at T + 1
pred.x <- PredPdf(object = fit, x = x, log = TRUE, do.its = FALSE)
```
Description

Method returning the Value-at-Risk and Expected-shortfall risk measures.

Usage

Risk(object, ...)

## S3 method for class 'MSGARCH_SPEC'
Risk(object, par, data, alpha = c(0.01, 0.05),
    nahead = 1L, do.es = TRUE, do.its = FALSE, do.cumulative = FALSE,
    ctr = list(), ...)

## S3 method for class 'MSGARCH_ML_FIT'
Risk(object, newdata = NULL, alpha = c(0.01, 0.05),
    do.es = TRUE, do.its = FALSE, nahead = 1L, do.cumulative = FALSE,
    ctr = list(), ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
Risk(object, newdata = NULL, alpha = c(0.01, 0.05),
    do.es = TRUE, do.its = FALSE, nahead = 1L,
    do.cumulative = FALSE, ctr = list(), ...)

Arguments

object Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.

... Not used. Other arguments to Risk.

par Vector (of size d) or matrix (of size nmcmc x d) of parameter estimates where d must have the same length as the default parameters of the specification.

data Vector (of size T) of observations.

alpha Vector (of size R) of Value-at-risk and Expected-shortfall levels.
( Default: alpha = c(0.01, 0.05) )

nahead Scalar indicating the number of step-ahead evaluation. (Default: nahead = 1L). Not used when do.its = TRUE as it only returns in-sample one-step ahead risk measures.

do.es Logical indicating if Expected-shortfall is also calculated. (Default: do.es = TRUE)

do.its Logical indicating if the in-sample risk estimators are returned. (Default: do.its = FALSE).

do.cumulative Logical indicating if the risk measures are computed on the cumulative simulations (typically log-returns, as they can be aggregated). Only available for do.its = FALSE. (Default: do.cumulative = FALSE)
cter A list of control parameters:

- \( \text{nmesh} \) (integer \( \geq 0 \)) : Number of points for density evaluation. (Default: \( \text{nmesh} = 1000L \))
- \( \text{nsim} \) (integer \( \geq 0 \)) : Number indicating the number of simulation done for estimation of the density at \( \text{nahead} > 1 \). (Default: \( \text{nsim} = 10000L \))

newdata Vector (of size \( T^* \)) of new observations. (Default \( \text{newdata} = \text{NULL} \))

Details

If a matrix of MCMC posterior draws is given, the Bayesian Value-at-Risk and Expected-shortfall are calculated. Two or more step ahead risk measures are estimated via simulation of \( \text{nsim} \) paths up to \( t = T + T^* + \text{nahead} \). If \( \text{do.its} = \text{FALSE} \), the risk estimators at \( t = T + T^* + 1, \ldots, t = T + T^* + \text{nahead} \) are computed. \( \text{do.cumulative} = \text{TRUE} \) indicate the function to compute the risk measure over aggregated period up to \( \text{nahead} \) period using the \text{cumsum} function on the simulated data.

Value

A list of class \text{MSGARCH_RISK} with the following elements:

- \( \text{VaR} \): If \( \text{do.its} = \text{FALSE} \): Value-at-Risk at \( t = T + T^* + 1, \ldots, t = T + T^* + \text{nahead} \) at the chosen levels (matrix of size \( \text{nahead} \times R \)).
  If \( \text{do.its} = \text{TRUE} \): In-sample Value-at-Risk at the chosen levels (Matrix of size \( (T + T^*) \times R \)).

- \( \text{ES} \): If \( \text{do.its} = \text{FALSE} \): Expected-shortfall at \( t = T + T^* + 1, \ldots, t = T + T^* + \text{nahead} \) at the chosen levels (matrix of size \( \text{nahead} \times R \)).
  If \( \text{do.its} = \text{TRUE} \): In-sample Expected-shortfall at the chosen levels (Matrix of size \( (T + T^*) \times R \)).

The \text{MSGARCH_RISK} contains the \text{plot} method. Note that the MCMC/Bayesian risk estimator can take long time to calculate depending on the size of the MCMC chain.

Examples

# create specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# risk from specification
par <- c(0.1, 0.1, 0.8, 0.2, 0.1, 0.8, 0.99, 0.01)
set.seed(1234)
risk <- Risk(object = spec, par = par, data = SMI, nahead = 5L)
head(risk)
plot(risk)

# risk from ML fit
fit <- FitML(spec = spec, data = SMI)
simulate.MSGARCH_SPEC

set.seed(1234)
risk <- Risk(object = fit, nahead = 5L)
head(risk)
plot(risk)

## Not run:
# risk from MCMC fit
set.seed(1234)
fit <- FitMCMC(spec = spec, data = SMI)
risk <- Risk(object = fit, nahead = 5L)
head(risk)
plot(risk)

## End(Not run)

---

**simulate.MSGARCH_SPEC**  
*Simulation of MSGARCH processes.*

**Description**

Method for simulating MSGARCH processes.

**Usage**

```r
## S3 method for class 'MSGARCH_SPEC'
simulate(object, nsim = 1L, seed = NULL,
          nahead = 1L, par = NULL, nburn = 500L, ...)
```

```r
## S3 method for class 'MSGARCH_ML_FIT'
simulate(object, nsim = 1L, seed = NULL,
          nahead = 1L, nburn = 500L, ...)
```

```r
## S3 method for class 'MSGARCH_MCMC_FIT'
simulate(object, nsim = 1L, seed = NULL,
          nahead = 1L, nburn = 500L, ...)
```

**Arguments**

- **object**: Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.
- **nsim**: Number of simulations. (Default: nsim = 1L)
- **seed**: Integer indicating if and how the random number generator should be initialized. If seed = NULL, the state of the random generator will not change. (Default: seed = NULL)
- **nahead**: Simulation length. (Default: nahead = 1L)
- **par**: Vector (of size d) or matrix (of size nahead x d) of parameter
nburn  Burnin period discarded (first simulation draws).
...  Not used. Other arguments to simulate.

Details
If a matrix of parameters estimates is provided, nsim simulations will be done for each row.

Value
A list of class MSGARCH_SIM with the following elements:

- **draw**: Matrix (of size nahead x nsim) of simulated draws.
- **state**: Matrix (of size nahead x nsim) of simulated states.
- **CondVol**: Array (of size nahead x nsim x K) of simulated conditional volatilities.

The MSGARCH_SIM class contains the plot method.

Examples

```r
# create specification
spec <- CreateSpec()

# simulation from specification
par <- c(0.1, 0.1, 0.8, 0.2, 0.1, 0.8, 0.99, 0.01)
set.seed(1234)
sim <- simulate(object = spec, nsim = 1L, nahead = 1000L,
                 nburnin = 500L, par = par)
head(sim)
plot(sim)

# load data
data("SMI", package = "MSGARCH")

# simulation from ML fit
fit <- FitML(spec = spec, data = SMI)
set.seed(1234)
sim <- simulate(object = fit, nsim = 1L, nahead = 1000L,
                 nburnin = 500L)
head(sim)
plot(sim)

## Not run:
# simulation from MCMC fit
fit <- FitMCMC(spec = spec, data = SMI)
set.seed(1234)
sim <- simulate(object = fit, nahead = 100L, nburnin = 500L)
head(sim)
plot(sim)

## End(Not run)
```
SMI

Swiss market index dataset

Description

See Mullen et al. (2011) for a description of this dataset.

Usage

data("SMI")

Format

zoo object containing 2,500 observations ranging from 1990-11-12 to 2000-10-20.

Source

DEoptim package

References


State

State probabilities.

Description

Method returning the filtered, predictive, and smoothed probabilities of the states, and the most probable path computed with the Viterbi algorithm.

Usage

State(object, ...)

## S3 method for class 'MSGARCH_SPEC'
State(object, par, data, ...)

## S3 method for class 'MSGARCH_ML_FIT'
State(object, newdata = NULL, ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
State(object, newdata = NULL, ...)
Arguments

object  Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH ML FIT created with FitML or MSGARCH MCMC FIT created with FitMCMC.

...  Not used. Other arguments to State.

par  Vector (of size d) or matrix (of size \( nmcmc \times d \)) of parameter estimates where \( d \) must have the same length as the default parameters of the specification.

data  Vector (of size T) of observations.

newdata  Vector (of size \( T^* \)) of new observations. (Default \( newdata = \text{NULL} \))

Details

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

Value

A list of class MSGARCH_PSTATE with the following elements:

- FiltProb: Filtered probabilities (array of size \((T + T^*) \times (nmcmc \text{ or } 1) \times K\)).
- PredProb: Predictive probabilities (array of size \((T + T^* + 1) \times (nmcmc \text{ or } 1) \times K\)).
- SmoothProb: Smoothed probabilities (array of size \((T + T^* + 1) \times (nmcmc \text{ or } 1) \times K\)).
- Viterbi: Most likely path (matrix of size \((T + T^*) \times (nmcmc \text{ or } 1)\)).

The class MSGARCH_PSTATE contains the plot method. The plot method contains as input type.prob which is one of "filtered", "predictive", "smoothed", "viterbi". (Default: type.prob = "smoothed")

Examples

```r
# create specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# state from specification
par <- c(0.1, 0.1, 0.8, 0.2, 0.1, 0.8, 0.99, 0.01)
state <- State(object = spec, par = par, data = SMI)
plot(state, type.prob = "filtered")

# state from ML fit
fit <- FitML(spec = spec, data = SMI)
state <- State(object = fit)
plot(state, type.prob = "smoothed")

## Not run:
# state from MCMC fit
set.seed(1234)
fit <- FitMCMC(spec = spec, data = SMI)
```
state <- State(object = fit)
plot(state, type.prob = "smoothed")

## End(Not run)

TransMat

Description

Method returning the transition matrix.

Usage

TransMat(object, ...)

## S3 method for class 'MSGARCH_SPEC'
TransMat(object, par = NULL, nahead = 1L, ...)

## S3 method for class 'MSGARCH_ML_FIT'
TransMat(object, nahead = 1L, ...)

Arguments

<table>
<thead>
<tr>
<th>argument</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML.</td>
</tr>
<tr>
<td>...</td>
<td>Not used. Other arguments to TransMat.</td>
</tr>
<tr>
<td>par</td>
<td>Vector (of size 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.</td>
</tr>
<tr>
<td>nahead</td>
<td>Number of steps ahead. (Default: nahead = 1L)</td>
</tr>
</tbody>
</table>

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 of GARCH model. The row indicates the starting states while the columns indicates the transition states.

Examples

# create model specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# fit the model on the data by ML
fit <- FitML(spec = spec, data = SMI)
# Extract the transition matrix 10 steps ahead
trans.mat <- TransMat(fit, nahead = 10)
print(trans.mat)

## UncVol

### Description
Method returning the unconditional volatility of the process.

### Usage

```r
UncVol(object, ...)  
## S3 method for class 'MSGARCH_SPEC'
UncVol(object, par = NULL, ctr = list(), ...)

## S3 method for class 'MSGARCH_ML_FIT'
UncVol(object, ctr = list(), ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
UncVol(object, ctr = list(), ...)
```

### Arguments

- `object` Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.
- `...` Not used. Other arguments to UncVol.
- `par` Vector (of size d) or matrix (of size nmc x d) of parameter estimates where d must have the same length as the default parameters of the specification.
- `ctr` A list of control parameters:
  - `nsim` (integer >= 0): Number of simulations used for the estimation of the unconditional volatility. (Default: `nsim = 250L`)
  - `nahead` (integer >= 0): Number of step ahead performed to estimate the unconditional volatility. (Default: `nahead = 5000L`)
  - `nburn` (integer >= 0): Number of discarded step to estimate the unconditional volatility. (Default: `nburn = 1000L`)

### Details
If a matrix of MCMC posterior draws is given, the Bayesian unconditional volatility is calculated. The unconditional volatility is estimated by first simulating `nsim` paths up to `nburn + nahead`, calculating a forecast of the conditional volatility at each step ahead, discarding the first `nburn` step ahead conditional volatilities forecasts, and computing the mean of the remaining `nahead - nburn`
Volatility filtering.

### Description
Method returning the in-sample conditional volatility.

### Usage

```r
Volatility(object, ...)  
```

```r  
## S3 method for class 'MSGARCH_SPEC'
Volatility(object, par, data, ...)  
```

```r  
## S3 method for class 'MSGARCH_ML_FIT'
```
Volatility(object, newdata = NULL, ...)

## S3 method for class 'MSGARCH_MCMC_FIT'
Volatility(object, newdata = NULL, ...)

**Arguments**

- **object**: Model specification of class MSGARCH_SPEC created with CreateSpec or fit object of type MSGARCH_ML_FIT created with FitML or MSGARCH_MCMC_FIT created with FitMCMC.
- **...**: Not used. Other arguments to Volatility.
- **par**: Vector (of size d) or matrix (of size nmc x d) of parameter estimates where d must have the same length as the default parameters of the specification.
- **data**: Vector (of size T) of observations.
- **newdata**: Vector (of size T*) of new observations. (Default newdata = NULL)

**Details**

If a matrix of MCMC posterior draws is given, the Bayesian predictive conditional volatility is calculated.

**Value**

In-sample conditional volatility (vector of size T + T*) of class MSGARCH_CONDVOL. The MSGARCH_CONDVOL class contains the plot method.

**Examples**

```r
# create specification
spec <- CreateSpec()

# load data
data("SMI", package = "MSGARCH")

# in-sample volatility from specification
par <- c(0.1, 0.1, 0.8, 0.2, 0.1, 0.8, 0.99, 0.01)
vol <- Volatility(object = spec, par = par, data = SMI)
head(vol)
plot(vol)

# in-sample volatility from ML fit
fit <- FitML(spec = spec, data = SMI)
vol <- Volatility(object = fit)
head(vol)
plot(vol)

## Not run:
# in-sample volatility from MCMC fit
set.seed(1234)
fit <- FitMCMC(spec = spec, data = SMI)
```
Volatility

```r
vol <- Volatility(object = fit)
head(vol)
plot(vol)

## End(Not run)
```
Index

*Topic datasets
  dem2gbp, 7
  SMI, 25

CreateSpec, 3, 10, 11, 13–15, 17, 19, 21, 23, 26–28, 30

  dem2gbp, 7
  DIC, 8, 11

ExtractStateFit, 9, 11, 14

FitMCMC, 5, 6, 8, 9, 10, 15, 17, 19, 21, 23, 26, 28, 30
  FitML, 6, 9, 13, 15, 17, 19, 21, 23, 26–28, 30

MSGARCH (MSGARCH-package), 2
  MSGARCH-package, 2

PIT, 6, 11, 14, 15
  predict.MSGARCH_MCMC_FIT
    (predict.MSGARCH_SPEC), 17
  predict.MSGARCH_ML_FIT
    (predict.MSGARCH_SPEC), 17
  predict.MSGARCH_SPEC, 17
  PredPdf, 6, 11, 14, 19

Risk, 6, 11, 14, 21

simulate.MSGARCH_MCMC_FIT
  (simulate.MSGARCH_SPEC), 23
simulate.MSGARCH_ML_FIT
  (simulate.MSGARCH_SPEC), 23
simulate.MSGARCH_SPEC, 23
  SMI, 25
  State, 6, 11, 14, 25

TransMat, 27

UncVol, 6, 11, 14, 28

Volatility, 6, 11, 14, 29