Package 'BEKKs'

November 8, 2022

Title Multivariate Conditional Volatility Modelling and Forecasting

Version 1.4.0

Description Methods and tools for estimating, simulating and forecasting of so-called BEKK-models (named after Baba, Engle, Kraft and Kroner) based on the fast Berndt–Hall–Hall–Hausman (BHHH) algorithm described in Hafner and Herwartz (2008) <doi:10.1007/s00184-007-0130-y>.

Depends R (>= 3.5.0)

Imports Rcpp, reshape2, ggplot2, mathjaxr, gridExtra, grid, ggfortify, parallel, xts, stats, future, forecast, future.apply, GAS, ks, lubridate, utils, pbapply, numDeriv, moments

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

SystemRequirements C++11

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat (>= 2.1.0)

RdMacros mathjaxr

RoxygenNote 7.1.2

R topics documented:

backtest *Backtesting via Value-at-Risk (VaR)*

Description

Method for backtesting a model obtained from bekk_fit in terms of VaR-forcasting accuracy using a rolling window approach.

Usage

```
backtest(
  x,
  window_length = 1000,
  p = 0.99,
  portfolio_weights = NULL,
  n.ahead = 1,
  distribution = "empirical",
  nc = 1
)
```

Arguments

x	An object of class "bekkFit" from the function bekk_fit.
window_length	An integer specifying the length of the rolling window.
p	A numerical value that determines the confidence level. The default value is set at 0.99 in accordance with the Basel Regulation.
portfolio_weights	A vector determining the portfolio weights to calculate the portfolio VaR. If set to "NULL", the univariate VaR for each series are calculated.
n.ahead	Number of periods to predict conditional volatility. Default is a one-period ahead forecast.
distribution	A character string determining the assumed distribution of the residuals. Implemented are "normal", "empirical" and "t". The default is assuming the empirical distribution of the residuals.
nc	Number of cores to be used for parallel computation.

Value

Returns a S3 class "backtest" object containing the VaR forecast, out-of-sample returns and backtest statistics according to the R-package "GAS". conf

Examples

```
data(StocksBonds)
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

# backtesting
x2 <- backtest(x1, window_length = 6000, n.ahead = 1, nc = 1)
```

```
plot(x2)
# backtesting using 5 day-ahead forecasts
x3 <- backtest(x1, window_length = 6000, n.ahead = 5, nc = 1)
plot(x3)
# backtesting using 20 day-ahead forecasts and portfolio
x4 <- backtest(x1, window_length = 6000, portfolio_weights = c(0.5,0.5), n.ahead = 20, nc = 1)
plot(x4)
```

BEKKs *BEKKs: Volatility modelling*

Description

This package implements estimation, simulation and forecasting techniques for conditional volatility modelling using the BEKK model. The full BEKK(1,1,1) model of Engle and Kroner (1995)

$$H_t = CC' + A' r_{t-1} r_{t-1}' A + G' H_{t-1} G,$$

the asymmetric extensions of Kroner and Ng (1998) and Grier et. al. (2004)

$$H_t = CC' + A' r_{t-1} r_{t-1}' A + B' \gamma_{t-1} \gamma_{t-1}' B + G' H_{t-1} G$$

with

$$\gamma_t = r_t I\left(r_t < 0\right)$$

are implemented. Moreover, the diagonal BEKK, where the parameter matrices A, B and G are reduced to diagonal matrices and the scalar BEKK model of Ding and Engle (2001)

$$H_t = CC' + a r_{t-1} r_{t-1}' + g H_{t-1},$$

where a and g are scalar parameters and are implemented to allow faster but less flexible estimation in higher dimensions.

Details

The main functions are:

bekk_spec Specifies the model type to be estimated,

- bekk_fit Estimates a BEKK(1,1,1) model of a given series and specification object bekk_spec,
- simulate Simulates a BEKK(1,1,1) process using either a bekk_fit or bekk_spec object,
-

predict Forecasts conditional volatility using a bekk_fit object,

- VaR Estimates (portfolio) Value-at-Risk using a fitted BEKK(1,1,1) model.
- backtest Uses estimated (portfolio) Value-at-Risk of a fitted BEKK(1,1,1) model to backtest the risk-forecasting accura
- virf Calculaes volatility impulse response functions for fitted symmetric BEKK(1,1,1) models as described by Hafner a

Author(s)

- Markus J. Fülle <fuelle@uni-goettingen.de>
- Helmut Herwartz <hherwartz@uni-goettingen.de>
- Alexander Lange <alexander.lange@uni-goettingen.de>
- Christian M. Hafner <christian.hafner@uclouvain.be>

References

Engle, R. F. and K. F. Kroner (1995). Multivariate simultaneous generalized arch. Econometric Theory 11(1),122-150.

Kroner, K. F. and V. K. Ng (1998). Modeling asymmetric comovements of asset returns. Review of Financial Studies 11(4), 817-44.

Ding, Zhuanxin and Engle, Robert F (2001). Large scale conditional covariance matrix modeling, estimation and testing. NYU working paper No. Fin-01-029.

Grier, K. B., Olan T. Henry, N. Olekalns, and K. Shields (2004). The asymmetric effects of uncertainty on inflation and output growth. Journal of Applied Econometrics 19(5), 551-565.

Hafner CM, Herwartz H (2006). Volatility impulse responses for multivariate GARCH models: An exchange rate illustration. Journal of International Money and Finance,25,719-740.

---

bekk_fit *Estimating multivariate BEKK-type volatility models*

Description

Method for fitting a variety of N-dimensional BEKK models.

Usage

```
bekk_fit(spec, data, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-09)
```

Arguments

spec	An object of class "bekkSpec" from function bekk_spec.
data	A multivariate data object. Can be a numeric matrix or ts/xts/zoo object.
QML_t_ratios	Logical. If QML_t_ratios = 'TRUE', the t-ratios of the BEKK parameter matrices are exactly calculated via second order derivatives.
max_iter	Maximum number of BHHH algorithm iterations.
crit	Determines the precision of the BHHH algorithm.

Details

The BEKK optimization routine is based on the Berndt–Hall–Hall–Hausman (BHHH) algorithm and is inspired by the study of Hafner and Herwartz (2008). The authors provide analytical formulas for the score and Hessian of several MGARCH models in a QML framework and show that analytical derivations significantly outperform numerical methods.

Value

Returns a S3 class "bekkFit" object containing the estimated parameters, t-values, volatility process of the model defined by the BEKK_spec object.

References

Hafner and Herwartz (2008). Analytical quasi maximum likelihood inference in multivariate volatility models. Metrika, 67, 219-239.

Examples

```
data(StocksBonds)

# Fitting a symmetric BEKK model
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

summary(x1)

plot(x1)

# Fitting an asymmetric BEKK model
obj_spec <- bekk_spec(model = list(type = "bekk", asymmetric = TRUE))
x1 <- bekk_fit(obj_spec, StocksBonds)

summary(x1)

plot(x1)

# Fitting a symmetric diagonal BEKK model
obj_spec <- bekk_spec(model = list(type = "dbekk", asymmetric = FALSE))
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

summary(x1)

plot(x1)
```

```
# Fitting a symmetric scalar BEKK model
obj_spec <- bekk_spec(model = list(type = "sbekk", asymmetric = FALSE))
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

summary(x1)

plot(x1)
```

---

bekk_spec *BEKK specification method*

Description

Method for creating a N-dimensional BEKK model specification object prior to fitting and/or simulating.

Usage

```
bekk_spec(
  model = list(type = "bekk", asymmetric = FALSE),
  init_values = NULL,
  signs = NULL,
  N = NULL
)
```

Arguments

model	A list containing the model type specification: Either "bekk" "dbekk" or "sbekk". Moreover it can be specified whether the model should be estimated allowing for asymmetric volatility structure.
init_values	initial values for bekk_fit during BHHH algorithm. It can be either a numerical vector of suitable dimension, or a character vector i.e. "random" to use a random starting value generator (set a seed in advance for reproducible results), or "simple" for relying on a simple initial values generator based on typical values for BEKK parameter found in the literature. If the object from this function is passed to simulate, "init_values" are used as parameters for data generating process.
signs	An N-dimensional vector consisting of "1" or "-1" to indicate the asymmetric effects to be considered. Setting the i-th element of the vector to "1" or "-1" means that the model takes into account additional volatility if the returns of the i-th column in the data matrix are either positive or negative. If "asymmetric = TRUE", the default is set to "rep(-1, N)" i.e. it is assumed that excess volatility occurs for all series if the returns are negative.
N	Integer specifying the dimension of the BEKK model. Only relevant when this object of class "bekkSpec"" is used for simulating BEKK processes by applying it to simulate.

Value

Returns a S3 class "bekkSpec" object containing the specifications of the model to be estimated.

`GoldStocksBonds` *Gold stock and Bond returns*

Description

Trivariate data set consisting of daily gold, S&P 500 and U.S. Treasury Bond Future returns from October 1991 to October 2021.

Usage

```
data("GoldStocksBonds")
```

Format

A data frame with 7346 observations on the following 3 variables.

Gold a numeric vector

S&P 500 a numeric vector

US Treasury Bond Future a numeric vector

Source

Yahoo Finance.

Examples

```
data(GoldStocksBonds)
## maybe str(GoldStocksBonds) ; plot(GoldStocksBonds) ...
```

`logLik.bekkFit` *bekkFit method*

Description

Generic 'bekkFit' methods. More details on 'bekkFit' are described in bekk_fit

Usage

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

## S3 method for class 'bekkFit'
AIC(object, ..., k = 2)

## S3 method for class 'bekkFit'
BIC(object, ...)

## S3 method for class 'bekkFit'
print(x, ...)

## S3 method for class 'bekkFit'
residuals(object, ...)
```

Arguments

object	An object of class "bekkFit" from function bekk_fit.
...	Further arguments to be passed to and from other methods.
k	Numeric value, the penalty per parameter to be used; the default k = 2 is the classical AIC.
x	An object of class "bekkFit" from function bekk_fit.

Examples

```
data(StocksBonds)

# Fitting a symmetric BEKK model
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

AIC(x1)
```

---

portmanteau.test *Performing a Portmanteau test checking for remaining correlation in the empirical co-variances of the estimated BEKK residuals.*

Description

Method for a Portmanteau test of the null hypothesis of no remaining correlation in the co-variances of the estimated BEKK residuals.

Usage

```
portmanteau.test(x, lags = 5)
```

Arguments

x	An object of class "bekkFit" from function bekk_fit.
lags	An integer defining the lag length.

Details

Here, the multivariate Portmanteau test of Hosking (1980) is implemented.

Value

Returns an Object of class "htest" containing the p-value and test statistic.

References

J. R. M. Hosking (1980). The Multivariate Portmanteau Statistic, Journal of the American Statistical Association, 75:371, 602-608.

---

predict *Forecasting conditional volatilities with BEKK models*

Description

Method for predicting a N-dimensional BEKK covariances.

Usage

```
predict(x, n.ahead = 1, ci = 0.95)
```

Arguments

x	A fitted bekk model of class bekk from the bekk_fit function
n.ahead	Number of periods to forecast conditional volatility. Default is a one-period ahead forecast.
ci	Floating point in [0,1] defining the niveau for confidence bands of the conditional volatility forecast. Default is 95 per cent niveau confidence bands.

Value

Returns a S3 class "bekkForecast" object containing the conditional volatility forecasts and respective confindence bands.

Examples

```
#'
data(StocksBonds)
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

x2 <- predict(x1, n.ahead = 1)
```

simulate *Simulating BEKK models*

Description

Method for simulating a N-dimensional BEKK model.

Usage

```
simulate(spec, nobs)
```

Arguments

spec	A spec object of class "bekkSpec" from the function bekk_spec or a fitted bekk model of class "bekkFit" from the bekk_fit function
nobs	Number of observations of the simulated sample

Value

Returns a simulated time series S3 class object using the parameters of passed "bekkSpec" or "bekkFit".

Examples

```
# Simulate a BEKK with estimated parameter
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds)

x2 <- simulate(x1, 3000)

plot(x2)
```

StocksBonds *Daily stock and Bond returns*

Description

Bivariate data set consisting of daily S&P 500 bond and MSCI World returns from December 1995 to December 2019.

Usage

```
data("StocksBonds")
```

Format

A data frame with 6073 observations on the following 2 variables.

S&P 500 Bonds a numeric vector

MSCI World a numeric vector

Source

Yahoo Finance.

Examples

```
data(StocksBonds)
## maybe str(StocksBonds) ; plot(StocksBonds) ...
```

VaR *Calculating Value-at-Risk (VaR)*

Description

Method for calculating VaR from estimated covariance processes (bekk_fit) or predicted covariances (forecast).

Usage

```
VaR(x, p = 0.99, portfolio_weights = NULL, distribution = "empirical")
```

Arguments

x	An object of class "bekkFit" from the function bekk_fit or an object of class "bekkForecast" from the function predict.
p	A numerical value that determines the confidence level. The default value is set at 0.99 in accordance with the Basel Regulation.
portfolio_weights	A vector determining the portfolio weights to calculate the portfolio VaR. If set to "NULL", the univariate VaR for each series are calculated.
distribution	A character string determining the assumed distribution of the residuals. Implemented are "normal", "empirical" and "t". The default is using the empirical distribution of the residuals.

Value

Returns a S3 class "var" object containing the VaR forecast and respective confidence bands.

Examples

```
data(StocksBonds)
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

# single VaRs of series
x2 <- VaR(x1, distribution="normal")
plot(x2)

# VaR of equally-weighted portfolio
portfolio_weights <- c(0.5, 0.5)
x3 <- VaR(x1, portfolio_weights = portfolio_weights)
plot(x3)

# VaR of traditional 30/70 weighted bond and stock portfolio
portfolio_weights <- c(0.3, 0.7)
x4 <- VaR(x1, portfolio_weights = portfolio_weights)
plot(x4)
```

`virf` *Estimating multivariate volatility impulse response functions (VIRF) for BEKK models*

Description

Method for estimating VIRFs of N-dimensional BEKK models. Currently, only VIRFs for symmetric BEKK models are implemented.

Usage

```
virf(
  x,
  time = 1,
  q = 0.05,
  index_series = 1,
  n.ahead = 10,
  ci = 0.9,
  time_shock = FALSE
)
```

Arguments

`x`	An object of class "bekkfit" from function bekk_fit.
`time`	Time instance to calculate VIRFs for.
`q`	A number specifying the quantile to be considered for a shock on which basis the VIRFs are generated.
`index_series`	An integer defining the number of series for which a shock is assumed.

n.ahead	An integer defining the number periods for which the VIRFs are generated.
ci	A number defining the confidence level for the confidence bands.
time_shock	Boolean indicating if the estimated residuals at date specified by "time" shall be used as a shock.

Value

Returns an object of class "virf".

References

Hafner CM, Herwartz H (2006). Volatility impulse responses for multivariate GARCH models: An exchange rate illustration. Journal of International Money and Finance,25,719–740.

Examples

```
data(StocksBonds)
obj_spec <- bekk_spec()
x1 <- bekk_fit(obj_spec, StocksBonds, QML_t_ratios = FALSE, max_iter = 50, crit = 1e-9)

# 250 day ahead VIRFs and 90% CI for a Shock in the 1% quantile  of Bonds (i.e. series=2)
# shock is supposed to occur at day 500
x2 <- virf(x1, time = 500, q = 0.01, index_series=2, n.ahead = 500, ci = 0.90)
plot(x2)
```

Index