Package ‘quarks’

March 31, 2022

Type Package

Title Simple Methods for Calculating and Backtesting Value at Risk and Expected Shortfall

Version 1.1.0


License GPL-3

Depends R (>= 2.10)

Imports graphics, progress, rugarch, smoots, stats

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Sebastian Letmathe [aut, cre] (Paderborn University, Germany)

Maintainer Sebastian Letmathe <sebastian.letmathe@uni-paderborn.de>

Repository CRAN

Date/Publication 2022-03-30 23:00:02 UTC

R topics documented:

cvgtest ................................................................. 2
DAX .................................................................. 4
DJI .................................................................. 4
Description

The conditional (Kupiec, 1995), the unconditional coverage test (Christoffersen, 1998) and the independence test (Christoffersen, 1998) of the Value-at-Risk (VaR) are applied.

Usage

cvgtest(obj = list(loss = NULL, VaR = NULL, p = NULL))

Arguments

obj a list that contains the following elements:

loss a numeric vector that contains the values of a loss series ordered from past to present; is set to NULL by default.

VaR a numeric vector that contains the estimated values of the VaR for the same time points of the loss series loss; is set to NULL by default.

p a numeric vector with one element; defines the probability p stated in the null hypotheses of the coverage tests (see the section Details for more information); is set to NULL by default.

Please note that a list returned by the rollcast function can be directly passed to cvgtest.

Details

The function needs three inputs: the out-of-sample loss series obj$loss, the corresponding estimated obj$VaR series and the coverage level obj$p, for which the VaR has been calculated. If an object returned by this function is entered into the R console, a detailed overview of the test results is printed.
Value

A list of class quarks with the following four elements:

- \( p \)  probability \( p \) stated in the null hypotheses of the coverage tests
- \( p.\text{uc} \)  the p-value of the unconditional coverage test
- \( p.\text{cc} \)  the p-value of the conditional coverage test
- \( p.\text{ind} \)  the p-value of the independence test

- **model**  selected model for estimation; only available if a list returned by the rollcast() is passed to cvgtest
- **method**  selected method for estimation; only available if a list returned by the rollcast() is passed to cvgtest

References


'@details With this function, the conditional and the unconditional coverage tests introduced by Kupiec (1995) and Christoffersen (1998) can be applied. Given a return series \( r_t \) with \( n \) observations, divide the series into \( n - K \) in-sample and \( K \) out-of-sample observations, fit a model to the in-sample data and obtain rolling one-step forecasts of the VaR for the out-of-sample time points. Define

\[
I_t = 1,
\]

if \(-r_t > \hat{\text{VaR}}_t(\alpha)\) or

\[
I_t = 0,
\]

otherwise, for \( t = n + 1, n + 2, \ldots, n + K \) as the hit sequence, where \( \alpha \) is the confidence level for the VaR (often \( \alpha = 0.95 \) or \( \alpha = 0.99 \)). Furthermore, denote \( p = \alpha \) and let \( w \) be the actual covered proportion of losses in the data.

Examples

```r
cvgest <- DAX$price.close returns <- diff(log(prices)) n <- length(returns) nout <- 250 # number of obs. for out-of-sample forecasting nwin <- 500 # window size for rolling forecasts results <- rollcast(x = returns, p = 0.975, method = 'age', nout = nout, nwin = nwin) cvgtest(results)
```
DAX

*German Stock Market Index (DAX) Financial Time Series Data*

**Description**

A dataset that contains the daily financial data of the DAX from 2000 to December 2021 (currency in EUR).

**Usage**

DAX

**Format**

A data frame with 5582 rows and 10 variables:

- **price.open** opening price (daily)
- **price.high** highest price (daily)
- **price.low** lowest price (daily)
- **price.close** closing price (daily)
- **volume** trading volume
- **price.adjusted** adjusted closing price (daily)
- **ref.date** date in format YY-MM-DD
- **ticker** ticker symbol
- **ret.adjusted.prices** returns obtained from the adj. closing prices
- **ret.closing.prices** returns obtained from the closing prices

**Source**

The data was obtained from Yahoo Finance.

DJI

*Dow Jones Industrial Average (DJI) Financial Time Series Data*

**Description**

A dataset that contains the daily financial data of the DJI from 2000 to December 2021 (currency in EUR).

**Usage**

DJI
ewma

Format

A data frame with 5535 rows and 10 variables:

- **price.open** opening price (daily)
- **price.high** highest price (daily)
- **price.low** lowest price (daily)
- **price.close** closing price (daily)
- **volume** trading volume
- **price.adjusted** adjusted closing price (daily)
- **ref.date** date in format YY-MM-DD
- **ticker** ticker symbol
- **ret.adjusted.prices** returns obtained from the adj. closing prices
- **ret.closing.prices** returns obtained from the closing prices

Source

The data was obtained from Yahoo Finance.

ewma  
*Exponentially weighted moving average*

Description

Estimates volatility of a return series by means of an exponentially weighted moving average.

Usage

```r
ewma(x, lambda = 0.94)
```

Arguments

- **x** a numeric vector of asset returns
- **lambda** decay factor for the calculation of weights; default is 0.94

Value

Returns a numerical vector `vol` that contains the computed volatility.

Examples

```r
prices <- DAX$price.close
returns <- diff(log(prices))
date <- DAX$ref.date[-1]
cvar <- ewma(x = returns, lambda = 0.94)
csig <- sqrt(cvar)
plot(date, csig, type = 'l',
     main = 'conditional standard deviations for the DAX30 return series')
```
Filtered historical simulation

Description

Calculates univariate Value at Risk and Expected Shortfall (Conditional Value at Risk) by means of filtered historical simulation. Volatility can be estimated with an exponentially weighted moving average or a GARCH-type model.

Usage

fhs(x, p = 0.975, model = c("EWMA", "GARCH"), lambda = 0.94, nboot = NULL, ...)

Arguments

x  a numeric vector of asset returns
p  confidence level for VaR calculation; default is 0.975
model  model for estimating conditional volatility; options are 'EWMA' and 'GARCH'; if model = 'GARCH', additional arguments can be adjusted via ...; default is 'EWMA'
lambda  decay factor for the calculation of weights; default is 0.94
nboot  size of bootstrap sample; must be a single non-NA integer value with nboot > 0; default is NULL
...  additional arguments of the ugarchspec function from the rugarch-package; only applied if model = 'GARCH'; default settings for the arguments variance.model and mean.model are:
  variance.model = list(model = 'sGARCH', garchOrder = c(1, 1))
  mean.model = list(armaOrder = c(0, 0))

Value

Returns a list with the following elements:

VaR  Calculated Value at Risk
ES  Calculated Expected Shortfall (Conditional Value at Risk)
p  Confidence level for VaR calculation

garchmod  The model fit. Is the respective GARCH fit for model = "GARCH" (see rugarch documentation) and 'EWMA' for model = "EWMA"
Examples

```
prices <- DAX$price.close
returns <- diff(log(prices))
# volatility weighting via EWMA
ewma <- fhs(x = returns, p = 0.975, model = "EWMA", lambda = 0.94,
nboot = 10000)

# volatility weighting via GARCH
garch <- fhs(x = returns, p = 0.975, model = "GARCH", variance.model =
list(model = "sGARCH"), nboot = 10000)
```

---

**FTSE100**

*Financial Times Stock Exchange Index (FTSE) Financial Time Series Data*

---

**Description**

A dataset that contains the daily financial data of the FTSE from 2000 to December 2021 (currency in EUR).

**Usage**

```
FTSE100
```

**Format**

A data frame with 5558 rows and 10 variables:

- **price.open** opening price (daily)
- **price.high** highest price (daily)
- **price.low** lowest price (daily)
- **price.close** closing price (daily)
- **volume** trading volume
- **price.adjusted** adjusted closing price (daily)
- **ref.date** date in format YY-MM-DD
- **ticker** ticker symbol
- **ret.adjusted.prices** returns obtained from the adj. closing prices
- **ret.closing.prices** returns obtained from the closing prices

**Source**

The data was obtained from Yahoo Finance.
### Nonparametric calculation of univariate Value at Risk and Expected Shortfall

**Description**

Computes Value at Risk and Expected Shortfall (Conditional Value at Risk) by means of plain and age-weighted historical simulation.

**Usage**

\[
\text{hs}(x, p = 0.975, \text{method} = \text{c("age", "plain")}, \lambda = 0.98)
\]

**Arguments**

- **x**: a numeric vector of asset returns
- **p**: confidence level for VaR calculation; default is 0.975
- **method**: method to be used for calculation; default is 'plain'
- **lambda**: decay factor for the calculation of weights; default is 0.98

**Value**

Returns a list with the following elements:

- **VaR**: Calculated Value at Risk
- **ES**: Calculated Expected Shortfall (Conditional Value at Risk)
- **p**: Confidence level for VaR calculation

**Examples**

```r
prices <- DAX$price.close
returns <- diff(log(prices))
hs(x = returns, p = 0.975, method = 'plain')
hs(x = returns, p = 0.975, method = 'age', lambda = 0.98)
```

---

### Hang Seng Index (HSI) Financial Time Series Data

**Description**

A dataset that contains the daily financial data of the HSI from 2000 to December 2021 (currency in EUR).

**Usage**

```
HSI
```

lossfun

Format

A data frame with 5424 rows and 10 variables:

- **price.open** opening price (daily)
- **price.high** highest price (daily)
- **price.low** lowest price (daily)
- **price.close** closing price (daily)
- **volume** trading volume
- **price.adjusted** adjusted closing price (daily)
- **ref.date** date in format YY-MM-DD
- **ticker** ticker symbol
- **ret.adjusted.prices** returns obtained from the adj. closing prices
- **ret.closing.prices** returns obtained from the closing prices

Source

The data was obtained from Yahoo Finance.

---

### Description

This functions allows for the calculation of loss functions in order to assess the performance of models in regard to forecasting ES.

### Usage

```r
lossfun(obj = list(loss = NULL, ES = NULL), beta = 1e-04)
```

### Arguments

- **obj**
  - a list that contains the following elements:
    - **loss** a numeric vector that contains the values of a loss series ordered from past to present; is set to NULL by default
    - **ES** a numeric vector that contains the estimated values of the ES for the same time points of the loss series loss; is set to NULL by default
  - Please note that a list returned by the `rollcast` function can be directly passed to `lossfun`.

- **beta**
  - a single numeric value; a measure for the opportunity cost of capital; default is 1e-04.
Details

Given a negative return series \texttt{obj$loss}, the corresponding Expected Shortfall (ES) estimates \texttt{obj$ES} and a parameter \texttt{beta} that defines the opportunity cost of capital, four different definitions of loss functions are considered.

Value

an S3 class object, which is a list of

- \texttt{loss.func1} regulatory loss function
- \texttt{loss.func2} firm’s loss function following Sarma et al. (2003)
- \texttt{loss.func3} loss function following Abad et al. (2015)
- \texttt{loss.func4} Feng’s loss function; a compromise of regulatory and firm’s loss function

References


Examples

```r
prices <- DAX$price.close
returns <- diff(log(prices))
n <- length(returns)
nout <- 250 # number of obs. for out-of-sample forecasting
nwin <- 500 # window size for rolling forecasts
results <- rollcast(x = returns, p = 0.975, method = 'age', nout = nout, nwin = nwin)
loss <- -results$xout
ES <- results$ES
loss.data <- list(loss = loss, ES = ES)
lossfun(loss.data)

# directly passing the output object of 'rollcast()' to 'lossfun()
lossfun(results)
```

Description

A dataset that contains the daily financial data of the NIK from 2000 to December 2021 (currency in EUR).
Usage

NIK225

Format

A data frame with 5391 rows and 10 variables:

- **price.open** opening price (daily)
- **price.high** highest price (daily)
- **price.low** lowest price (daily)
- **price.close** closing price (daily)
- **volume** trading volume
- **price.adjusted** adjusted closing price (daily)
- **ref.date** date in format YY-MM-DD
- **ticker** ticker symbol
- **ret.adjusted.prices** returns obtained from the adj. closing prices
- **ret.closing.prices** returns obtained from the closing prices

Source

The data was obtained from Yahoo Finance.

---

**plop**

*Profit & Loss operator function*

Description

Calculates portfolio returns or losses by assigning weights

Usage

```
plop(x, wts = NULL, approxim = c(0, 1))
```

Arguments

- **x**
  - a numeric matrix of asset returns or losses
- **wts**
  - a numeric vector or matrix containing the portfolio weights; portfolio value is standardized to 1 on any observation unit; sum of weights should not exceed 1 (row-wise for matrices); by default the portfolio is equally weighted over time and across all assets; if a vector is passed to wts the portfolio is equally weighted over time
- **approxim**
  - controls if a first-order approximation for the calculation of returns or losses is used; default is 1 (first-order approximation is employed)
Value

Returns a list with the following elements:

pl  Weighted portfolio returns or losses
wts Portfolio weights

Examples

```r
# creating portfolio
portfol <- cbind(SP500$price.close, DJI$price.close)
returns <- apply(portfol, 2, function(x) diff(log(x)))
# defining weights and applying the P&L operator function
wts <- c(0.4, 0.6)
portret <- plop(returns, wts = wts, approxim = 1)
portloss <- plop(-returns, wts = wts, approxim = 1)
plot.ts(cbind(portret$pl, portloss$pl))
```

---

## S3 method for class 'quarks'

plot(x, ...)

### Arguments

- **x**
  - an input object of class quarks.
- **...**
  - additional arguments of the standard plot method.

### Value

None
Description

This function regulates how objects created by the package quarks are printed.

Usage

```r
## S3 method for class 'quarks'
print(x, ...)
```

Arguments

- `x`: an input object of class quarks.
- `...`: included for compatibility; additional arguments will however not affect the output.

Value

None

---

**rollcast**  
Rolling one-step ahead forecasts of Value at Risk and Expected Shortfall

Description

Computes rolling one-step ahead forecasts of Value at Risk and Expected Shortfall (Conditional Value at Risk) by means of plain historical simulation age- and volatility-weighted historical simulation as well as filtered historical simulation.

Usage

```r
rollcast(
  x, 
  p = 0.975, 
  model = c("EWMA", "GARCH"), 
  method = c("plain", "age", "vwhs", "fhs"), 
  lambda = c(0.94, 0.98), 
  nout = NULL, 
  nwin = NULL, 
  nboot = NULL, 
  smoothscale = c("none", "lpr", "auto"), 
  smoothopts = list(), 
  ...
)
```
Arguments

x  a numeric vector of asset returns

p  confidence level for VaR calculation; default is 0.975

model  model for estimating conditional volatility; options are 'EWMA' and 'GARCH'; if model = 'GARCH', additional arguments can be adjusted via ...; default is 'EWMA'

method  method to be used for calculation; default is 'plain'

lambda  decay factor for the calculation of weights; default is 0.98 for method = 'age' and 0.94 for method = 'vwhs' or method = 'fhs'

nout  number of out-of-sample observations; most recent observations are used; default is NULL

nwin  window size for rolling one-step forecasting; most recent observations before out-of-sample are used; default is NULL

nboot  size of bootstrap sample; must be a single non-NA integer value with nboot > 0; default is NULL

smoothscale  a character object; defines the smoothing approach for the unconditional variance from the logarithm of the squared centralized returns; for smoothscale = 'lpr', the unconditional variance is smoothed via the smoots::gsmooth() function from the smoots package; the bandwidth has to be chosen manually; otherwise the default is used; if smoothscale = 'auto', the function smoots::msmooth() is employed and the bandwidth is chosen automatically (data-driven); see the documentation of the smoots package for more information; is set to smoothscale = 'none' by default

smoothopts  additional arguments of smoots::gsmooth() and smoots::msmooth(); see the documentation of the smoots package for more information; is set to customized default settings

...  additional arguments of the ugarchspec function from the rugarch-package; only applied if model = 'GARCH'; default settings for the arguments variance.model and mean.model are:

  variance.model = list(model = 'sGARCH', garchOrder = c(1, 1))
  mean.model = list(armaOrder = c(0, 0))

Value

Returns a list with the following elements:

**VaR**  Numerical vector containing out-of-sample forecasts of Value at Risk

**ES**  Numerical vector containing out-of-sample forecasts of Expected Shortfall (Conditional Value at Risk)

**xout**  Numerical vector containing out-of-sample returns

**p**  Confidence level for VaR calculation

**model**  Model for estimating conditional volatility

**method**  Method to be used for calculation
**nout** Number of out-of-sample observations

**nwin** Window size for rolling one-step forecasting

**nboot** Size of bootstrap sample

### Examples

```r
prices <- DAX$price.close
returns <- diff(log(prices))
n <- length(returns)
nout <- 250  # number of obs. for out-of-sample forecasting
nwin <- 500  # window size for rolling forecasts

### Example 1 - plain historical simulation
results1 <- rollcast(x = returns, p = 0.975, method = 'plain', nout = nout, nwin = nwin)
matplot(1:nout, cbind(-results1$xout, results1$VaR, results1$ES),
     type = 'h',
     xlab = 'number of out-of-sample obs.', ylab = 'losses, VaR and ES',
     main = 'Plain HS - 97.5% VaR and ES for the DAX30 return series')

### Example 2 - age weighted historical simulation
results2 <- rollcast(x = returns, p = 0.975, method = 'age', nout = nout, nwin = nwin)
matplot(1:nout, cbind(-results2$xout, results2$VaR, results2$ES),
     type = 'h',
     xlab = 'number of out-of-sample obs.', ylab = 'losses, VaR and ES',
     main = 'Age weighted HS - 97.5% VaR and ES for the DAX30 return series')

### Example 3 - volatility weighted historical simulation - EWMA
results3 <- rollcast(x = returns, p = 0.975, model = 'EWMA', method = 'vwhs', nout = nout, nwin = nwin)
matplot(1:nout, cbind(-results3$xout, results3$VaR, results3$ES),
     type = 'h',
     xlab = 'number of out-of-sample obs.', ylab = 'losses, VaR and ES',
     main = 'Vol. weighted HS (EWMA) - 97.5% VaR and ES for the DAX30 return series')

### Example 4 - volatility weighted historical simulation - GARCH
results4 <- rollcast(x = returns, p = 0.975, model = 'GARCH', method = 'vwhs', nout = nout, nwin = nwin)
matplot(1:nout, cbind(-results4$xout, results4$VaR, results4$ES),
     type = 'h',
     xlab = 'number of out-of-sample obs.', ylab = 'losses, VaR and ES',
     main = 'Vol. weighted HS (GARCH) - 97.5% VaR and ES for the DAX30 return series')

### Example 5 - filtered historical simulation - EWMA
results5 <- rollcast(x = returns, p = 0.975, model = 'EWMA', method = 'fhs', nout = nout, nwin = nwin, nboot = 10000)
matplot(1:nout, cbind(-results5$xout, results5$VaR, results5$ES),
```
type = 'hll',
xlab = 'number of out-of-sample obs.', ylab = 'losses, VaR and ES',
main = 'Filtered HS (EWMA) - 97.5% VaR and ES for the DAX30 return series')

### Example 6 - filtered historical simulation - GARCH
results6 <- rollcast(x = returns, p = 0.975, model = 'GARCH',
method = 'fhs', nout = nout, nwin = nwin, nboot = 10000)
matplot(1:nout, cbind(-results6$xout, results6$VaR, results6$ES),
  type = 'hll',
xlab = 'number of out-of-sample obs.', ylab = 'losses, VaR and ES',
main = 'Filtered HS (GARCH) - 97.5% VaR and ES for the DAX30 return series')

---

**SP500**

*Standard and Poor's (SP500) Financial Time Series Data*

**Description**

A dataset that contains the daily financial data of the SP500 from 2000 to December 2021 (currency in EUR).

**Usage**

SP500

**Format**

A data frame with 5535 rows and 10 variables:

- `price.open` opening price (daily)
- `price.high` highest price (daily)
- `price.low` lowest price (daily)
- `price.close` closing price (daily)
- `volume` trading volume
- `price.adjusted` adjusted closing price (daily)
- `ref.date` date in format YY-MM-DD
- `ticker` ticker symbol
- `ret.adjusted.prices` returns obtained from the adj. closing prices
- `ret.closing.prices` returns obtained from the closing prices

**Source**

The data was obtained from Yahoo Finance.
Backtesting of Value-at-Risk via Traffic Light Test

Description

The Traffic Light Test, is applied to previously calculated Value-at-Risk series.

Usage

trftest(obj)

Arguments

obj A list returned by the rollcast function, that contains a Value-at-Risk series; any other list that follows the name conventions of the rollcast function can be used as well.

Details

This function uses an object returned by the rollcast function of the quarks package as an input for the function argument obj. A list with different elements, such as the cumulative probabilities for the VaR series within obj, is returned. Instead of the list, only the traffic light backtesting results are printed to the R console.

Value

A list of class quarks is returned with the following elements.

- **model** selected model for estimation
- **method** selected method for estimation
- **p_VaR** cumulative probability of observing the number of breaches or fewer for \((1 - p)100\%\)-VaR
- **pot_VaR** number of exceedances for \((1 - p)100\%\)-VaR
- **p** coverage level for \((1-p)100\%\) VaR

Examples

```r
prices <- DAX$price.close
returns <- diff(log(prices))
n <- length(returns)
nout <- 250 # number of obs. for out-of-sample forecasting
nwin <- 500 # window size for rolling forecasts
results <- rollcast(x = returns, p = 0.975, method = 'age', nout = nout, nwin = nwin)
trftest(results)
```
vwhs

Volatility weighted historical simulation

Description
Calculates univariate Value at Risk and Expected Shortfall (Conditional Value at Risk) by means of volatility weighted historical simulation. Volatility can be estimated with an exponentially weighted moving average or a GARCH-type model.

Usage
vwhs(x, p = 0.975, model = c("EWMA", "GARCH"), lambda = 0.94, ...)

Arguments
x a numeric vector of asset returns
p confidence level for VaR calculation; default is 0.975
model model for estimating conditional volatility; default is 'EWMA'
lambda decay factor for the calculation of weights; default is 0.94
... additional arguments of the ugarchspec function from the rugarch-package; the default settings for the arguments variance.model and mean.model are list(model = 'sGARCH', garchOrder = c(1,1)) and list(armaOrder = c(0,0)), respectively

Value
Returns a list with the following elements:

VaR Calculated Value at Risk
ES Calculated Expected Shortfall (Conditional Value at Risk)
p Confidence level for VaR calculation
garchmod The model fit. Is the respective GARCH fit for model = 'GARCH' (see rugarch documentation) and 'EWMA' for model = 'EWMA'

Examples
prices <- DAX$price.close
returns <- diff(log(prices))
# volatility weighting via EWMA
ewma <- vwhs(x = returns, p = 0.975, model = "EWMA", lambda = 0.94)

ewma

# volatility weighting via GARCH
garch <- vwhs(x = returns, p = 0.975, model = "GARCH", variance.model = list(model = "sGARCH"))
garch
Index

* datasets
  DAX, 4
  DJI, 4
  FTSE100, 7
  HSI, 8
  NIK225, 10
  SP500, 16

  cvgtest, 2
  DAX, 4
  DJI, 4

  ewma, 5

  fhs, 6
  FTSE100, 7

  hs, 8
  HSI, 8

  lossfun, 9

  NIK225, 10

  plop, 11
  plot.quarks, 12
  print.quarks, 13

  rollcast, 13

  SP500, 16

  trftest, 17

  vwhs, 18