Package ‘FRAPO’

February 19, 2015

Version 0.3-8
Date 2013-06-15
Title Financial Risk Modelling and Portfolio Optimisation with R
Depends R (>= 2.11), methods, quadprog, Rglpk, timeSeries
Suggests xts, zoo, Rsolnp
Description Accompanying package of the book 'Financial Risk Modelling and Portfolio Optimisation with R'. The data sets used in the book are contained in this package.
LazyData TRUE
License GPL (>= 2)
Author Bernhard Pfaff [aut, cre], Miguel Sousa Lobo [ctb] (SOCP), Lieven Vandenberghe [ctb] (SOCP), Stephen Boyd [ctb] (SOCP), Herve Lebret [ctb] (SOCP)
Maintainer Bernhard Pfaff <bernhard@pfaffikus.de>
Repository CRAN
Repository/R-Forge/Project frapo
Repository/R-Forge/Revision 23
Repository/R-Forge/DateTimeStamp 2013-06-15 10:12:00
Date/Publication 2013-06-19 00:34:43
NeedsCompilation yes

R topics documented:

BookEx ................................................................. 3
capser ............................................................... 4
DivMeasures .......................................................... 5
ESCBFX ................................................................. 7
EuroStoxx50 ........................................................... 8
FTSE100 ............................................................... 9
INDTRACK1 ............................................................ 10
INDTRACK2 ........................................................... 11
R topics documented:

INDTRACK3 ................................................. 12
INDTRACK4 ................................................. 13
INDTRACK5 ................................................. 14
INDTRACK6 ................................................. 15
MIBTEL ...................................................... 16
mrc ......................................................... 17
MultiAsset ................................................. 17
NASDAQ ................................................. 18
PAveDD ...................................................... 19
PCDaR ....................................................... 20
PERC ......................................................... 21
PERC2 ......................................................... 23
PGMV ......................................................... 24
plot-methods ............................................... 25
PMaxDD ...................................................... 26
PMD ......................................................... 27
PMinCDaR ................................................... 28
PMTD ......................................................... 29
PortAdd-class ............................................. 30
PortCdd-class ............................................. 31
PortDD-class ............................................. 32
PortMdd-class ............................................. 33
PortSol-class ............................................. 34
returnconvert ............................................. 35
returnsups ................................................. 36
Socp ......................................................... 37
SocpControl .............................................. 39
SocpPhase1 ............................................... 40
SocpPhase2 ............................................... 41
SP500 ....................................................... 42
sqrm ......................................................... 43
StockIndex ............................................... 44
StockIndexAdj ............................................ 45
StockIndexAdjD .......................................... 45
tdc ......................................................... 46
trdbilson ............................................... 47
trdbinary ............................................... 48
trdes ....................................................... 50
trdhp ....................................................... 51
trdsmo ..................................................... 52
trdwma ..................................................... 54

Index ...................................................... 56
Utility functions for handling book examples

Description

Utility functions for returning a list of the included examples and displaying, executing, saving and editing the example codes are provided.

Usage

listEx()
showEx(Example)
saveEx(Example)
editEx(Example, ...)
runEx(Example, ...)

Arguments

Example Character, the name of the example as contained in listEx().
...
Ellipsis argument. See details.

Details

The ellipsis arguments in the function editEx() are passed down to the function file.edit(). If the option editor is unset and/or a different editor shall be employed for opening the example code, then the ellipsis argument can be utilised by editor = "foo", wherey foo is the name of the editor to be used.
The ellipsis arguments in the function runEx() are passed down to the function source().

Value

listEx
Returns a character vector of the examples’ names.
showEx
Returns the example of of Example to the console.
saveEx
Returns a logical whether the saving of the R code example into the working directory was successful.
editEx
Opens a copy of the example code in an editor.
runEx
Executes the example code.

Author(s)

Bernhard Pfaff
See Also

`file.edit, source`

Examples

```r
## Not run:
listEx()
showEx(Example = "Part1Chapter3Ex2")
saveEx(Example = "Part1Chapter3Ex2")
runEx(Example = "Part1Chapter3Ex2", echo = TRUE)
editEx(Example = listEx()[1], editor = "emacs")

## End(Not run)
```

capser  

Capping a series to bounds

Description

The values of a series that are absolute greater than `min` and/or `max` are capped to these specified values.

Usage

capser(y, min, max)

Arguments

- `y` Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.
- `min` Numeric, minimum value for the series.
- `max` Numeric, maximim value for the series.

Value

An object of the same class as `y`, containing the truncated series.

Methods

- `y = "data.frame"` The calculation is applied per column of the data.frame and only if all columns are numeric.
- `y = "matrix"` The calculation is applied per column of the matrix.
- `y = "mts"` The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
- `y = "numeric"` Calculation of the es trend.
**DivMeasures**

The calculation is applied per column of the `timeSeries` object and an object of the same class is returned.

**y = "ts"** Calculation of the es trend. The attributes are preserved and an object of the same class is returned.

**y = "xts"** Calculation of the es trend. The attributes are preserved and an object of the same class is returned.

**y = "zoo"** Calculation of the es trend. The attributes are preserved and an object of the same class is returned.

**Author(s)**

Bernhard Pfaff

**See Also**

`trdbilson`, `trdbinary`, `trdes`, `trdhp`, `trdsma`, `trdwma`

**Examples**

```r
data(StockIndex)
y <- StockIndex[, "SP500"]
cs <- capser(y, min = 100, max = 200)
head(cs)
```

---

**DivMeasures**

**Diversification Measures**

**Description**

These functions compute the diversification ratio, the volatility weighted average correlation and concentration ratio of a portfolio.

**Usage**

```r
dr(weights, Sigma)
cr(weights, Sigma)
rhow(weights, Sigma)
```

**Arguments**

- **weights** Vector: portfolio weights.
- **Sigma** Matrix: Variance-covariance matrix of portfolio assets.
Details

The diversification ratio of a portfolio is defined as:

$$DR(\omega) = \sum_{i=1}^{N} \frac{\omega_i \sigma_i}{\sqrt{\omega^T \Sigma \omega}}$$

for a portfolio of \( N \) assets and \( \omega_i \) signify the weight of the i-th asset and \( \sigma_i \) its standard deviation and \( \Sigma \) the variance-covariance matrix of asset returns. The diversification ratio is therefore the weighted average of the assets’ volatilities divided by the portfolio volatility.

The concentration ration is defined as:

$$CR = \frac{\sum_{i=1}^{N} (\omega_i \sigma_i)^2}{(\sum_{i=1}^{N} \omega_i \sigma_i)^2}$$

and the volatility-weighted average correlation of the assets as:

$$\rho(\omega) = \frac{\sum_{i>j}^{N} (\omega_i \sigma_j \omega_j \sigma_i) \rho_{ij}}{\sum_{i>j}^{N} (\omega_i \sigma_j \omega_j \sigma_i)}$$

The following equation between these measures does exist:

$$DR(\omega) = \frac{1}{\sqrt{\rho(\omega)(1 - CR(\omega))} + CR(\omega)}$$

Value

numeric, the value of the diversification measure.

Author(s)

Bernhard Pfaff

References


See Also

PMD
Examples

data(MultiAsset)
Rets <- returnseries(MultiAsset, method = "discrete", trim = TRUE)
w <- Weights(PMO(Rets))
V <- cov(Rets)
DR <- dr(w, V)
CR <- cr(w, V)
Rhow <- rhow(w, V)
test <- 1 / sqrt(Rhow * (1 - CR) + CR)
all.equal(DR, test)

ESCBFX

Description

Daily spot rates of major currencies against the EUR.

Usage

data(ESCBFX)

Format

A data frame with 3,427 daily observations of the spot currency rates AUD, CAD, CHF, GBP, HKD, JPY and USD against EUR. The sample starts in 1999-01-04 and ends in 2012-04-04.

Details

The data has been retrieved from the Statistical Data Warehouse (SDW) Internet-Site of the ECB. In case of missing data entries due to holidays, the last observed data point has been carried forward.

Source

http://sdw.ecb.europa.eu

Examples

data(ESCBFX)
EuroStoxx50

EuroStoxx50  

Description
Weekly price data of 48 EURO STOXX 50 constituents.

Usage
data(EuroStoxx50)

Format
A data frame with 265 weekly observations of 48 members of the EURO STOXX 50 index. The sample starts at 2003-03-03 and ends in 2008-03-24.

Details
The data set was used in the reference below. The authors adjusted the price data for dividends and have removed stocks if two or more consecutive missing values were found. In the remaining cases the NA entries have been replaced by interpolated values.

Source
http://w3.uniroma1.it/Tardella/datasets.html
http://finance.yahoo.com/

References

Examples
data(EuroStoxx50)
Description

Weekly price data of 79 FTSE 100 constituents.

Usage

data(FTSE100)

Format

A data frame with 265 weekly observations of 79 members of the FTSE 100 index. The sample starts at 2003-03-03 and ends in 2008-03-24.

Details

The data set was used in the reference below. The authors adjusted the price data for dividends and have removed stocks if two or more consecutive missing values were found. In the remaining cases the NA entries have been replaced by interpolated values.

Source

http://w3.uniroma1.it/Tardella/datasets.html
http://finance.yahoo.com/

References


Examples

data(FTSE100)
**INDTRACK1:** Hang Seng Index and Constituents

**Description**
Weekly price data of the Hang Seng index and 31 constituents.

**Usage**
```r
data(INDTRACK1)
```

**Format**

**Details**
The data set was used in the first two references below. Stocks with missing values during the sample period have been discarded. The data was downloaded from DATASTREAM and has been anonymized. The first column refers to the index data itself. See the attached license file that is part of this package: 'BeasleyLicence'.

**Source**
http://people.brunel.ac.uk/~mastjjb/jeb/info.html
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/legal.html

**References**

**Examples**
```r
data(INDTRACK1)
```
INDTRACK2: DAX 100 Index and Constituents

Description

Weekly price data of the DAX 100 and 85 constituents.

Usage

data(INDTRACK2)

Format

A data frame with 291 weekly observations of the index and 85 members of the DAX 100 index. The sample starts in March 1991 and ends in September 1997.

Details

The data set was used in the first two references below. Stocks with missing values during the sample period have been discarded. The data was downloaded from DATASTREAM and has been anonymized. The first column refers to the index data itself. See the attached license file that is part of this package: 'BeasleyLicence'.

Source

http://people.brunel.ac.uk/~mastjjb/jeb/info.html
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/legal.html

References


Examples

data(INDTRACK2)
Description
Weekly price data of the FTSE 100 index and 89 constituents.

Usage

data(INDTRACK3)

Format
A data frame with 291 weekly observations of the index and 89 members of the FTSE 100 index. The sample starts in March 1991 and ends in September 1997.

Details
The data set was used in the first two references below. Stocks with missing values during the sample period have been discarded. The data was downloaded from DATASTREAM and has been anonymized. The first column refers to the index data itself. See the attached license file that is part of this package: 'BeasleyLicence'.

Source
http://people.brunel.ac.uk/~mastjjb/jeb/info.html
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/legal.html

References

Examples

data(INDTRACK3)
Description

Weekly price data of S&P 100 index and 98 constituents.

Usage

data(INTRACK4)

Format

A data frame with 291 weekly observations of the index 98 members of the S&P 100 index. The sample starts in March 1991 and ends in September 1997.

Details

The data set was used in the first two references below. Stocks with missing values during the sample period have been discarded. The data was downloaded from DATASTREAM and has been anonymized. The first column refers to the index data itself. See the attached license file that is part of this package: ‘BeasleyLicence’.

Source

http://people.brunel.ac.uk/~mastjjb/jeb/info.html
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/legal.html

References


Examples

data(INTRACK4)
**INDTRACK5: Nikkei 225 Index and Constituents**

**Description**

Weekly price data of Nikkei 225 index and 225 constituents.

**Usage**

data(INSTRACK5)

**Format**

A data frame with 291 weekly observations of the index and 225 members of the Nikkei 225 index. The sample starts in March 1991 and ends in September 1997.

**Details**

The data set was used in the first two references below. Stocks with missing values during the sample period have been discarded. The data was downloaded from DATASTREAM and has been anonymized. The first column refers to the index data itself. See the attached license file that is part of this package: 'BeasleyLicence'.

**Source**

http://people.brunel.ac.uk/~mastjjb/jeb/info.html  
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/legal.html

**References**


**Examples**

data(INSTRACK5)
Description

Weekly price data of S&P 500 index and 457 constituents.

Usage

data(INDTRACK6)

Format


Details

The data set was used in the first two references below. Stocks with missing values during the sample period have been discarded. The data was downloaded from DATASTREAM and has been anonymized. The first column refers to the index data itself. See the attached license file that is part of this package: ‘BeasleyLicence’.

Source

http://people.brunel.ac.uk/~mastjjb/jeb/info.html
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/legal.html

References


Examples

data(INDTRACK6)
Description

Weekly price data of 226 MIBTEL constituents.

Usage

data(MIBTEL)

Format

A data frame with 265 weekly observations of 226 members of the Milano Indice Borsa Telematica index. The sample starts at 2003-03-03 and ends in 2008-03-24.

Details

The data set was used in the reference below. The authors adjusted the price data for dividends and have removed stocks if two or more consecutive missing values were found. In the remaining cases the NA entries have been replaced by interpolated values.

Source

http://w3.uniroma1.it/Tardella/datasets.html
http://finance.yahoo.com/

References


Examples

data(MIBTEL)
**mrc**

*Marginal Contribution to Risk*

Description

This function returns the marginal contributions to portfolio risk, whereby the latter is defined in terms of the portfolio standard deviation.

Usage

```r
mrc(weights, Sigma, percentage = TRUE)
```

Arguments

- `weights`: Vector: portfolio weights.
- `Sigma`: Matrix: Variance-covariance matrix of portfolio assets.
- `percentage`: Logical, whether the marginal risk contributions shall be returned as percentages that sum to 100 (default) or as decimal numbers.

Details

The marginal contributions to risk are computed for a given dispersion matrix and weight vector.

Value

numeric, the marginal risk contributions of the portfolio’s asset.

Author(s)

Bernhard Pfaff

---

**MultiAsset**

*Multi Asset Index Data*

Description

Month-end price data of stock and bond indices and gold.

Usage

```r
data(MultiAsset)
```

Format

Details

The data set has been obtained from Yahoo Finance and hereby the unadjusted closing prices have been retrieved. If a month-end value was not reported, the value of the previous day has been used. The Yahoo mnemonics with the respective item description are listed below:

**GSPC** United States: S \& P 500 Index (Equity)

**RUA** United States: Russell 3000 Index (Equity)

**GDAXI** Germany: DAX (XETRA) Index (Equity)

**FTSE** United Kingdom: FTSE 100 Index (Equity)

**N225** Japan: Nikkei 225 Index (Equity)

**EEM** iShares: MSCI Emerging Markets Index (Equity)

**DJCBTI** United States: Dow Jones CBOT Treasury Index (Bonds)

**GREXP** Germany: REX-Performance Index (Bonds)

**BG05.L** United Kingdom: Gilt All Index (Bonds)

**GLD** United States: SPDR Gold Shares (Commodities)

Source

http://finance.yahoo.com/

Examples

```r
data(MultiAsset)
```

<table>
<thead>
<tr>
<th>NASDAQ</th>
<th>NASDAQ</th>
</tr>
</thead>
</table>

Description

Weekly price data of 2,196 NASDAQ constituents.

Usage

```r
data(NASDAQ)
```

Format

A data frame with 265 weekly observations of 2196 members of the NASDAQ index. The sample starts at 2003-03-03 and ends in 2008-03-24.

Details

The data set was used in the reference below. The authors adjusted the price data for dividends and have removed stocks if two or more consecutive missing values were found. In the remaining cases the NA entries have been replaced by interpolated values.
Source

http://w3.uniroma1.it/Tardella/datasets.html
http://finance.yahoo.com/

References


Examples

data(NASDAQ)

PAveDD  Portfolio optimisation with average draw down constraint

Description

This function returns the result of a long-only portfolio optimization whereby the portfolio’s (historic) average draw down is constrained to an upper limit.

Usage

PAveDD(PriceData, AveDD = 0.1, softBudget = FALSE, ...)

Arguments

PriceData  A rectangular array of price data.
AveDD     Numeric, the upper bound of the average portfolio draw down.
softBudget Logical, whether the budget constraint shall be implemented as a soft constraint, i.e. the sum of the weights can be less than one. The default is to use an equality constraint.
...       Arguments are passed down to Rglpk_solve_LP

Details

This function implements a long-only portfolio optimisation with an average draw down constraint (see references below). The problem can be stated in the form of a linear program and GLPK is used as solver.

Value

An object of formal class "PortAdd".
Note

A warning is issued in case the solver had exit status not equal to zero.

Author(s)

Bernhard Pfaff

References


See Also

"PortSol", "PortAdd", "PortDD", PMaxDD, PCDaR, PMinCDaR

Examples

```r
## Not run:
data(StockIndex)
popt <- PAveDD(PriceData = StockIndex, AveDD = 0.1, softBudget = TRUE)

## End(Not run)
```

PCDaR

Portfolio optimisation with conditional draw down at risk constraint

Description

This function returns the result of a long-only portfolio optimization whereby the portfolio’s (historic) conditional draw down at risk is constrained to an upper limit.

Usage

PCDaR(PriceData, alpha = 0.95, bound = 0.05, softBudget = FALSE, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PriceData</td>
<td>A rectangular array of price data.</td>
</tr>
<tr>
<td>alpha</td>
<td>Numeric, the confidence level for which the conditional draw down shall be</td>
</tr>
<tr>
<td></td>
<td>computed.</td>
</tr>
<tr>
<td>bound</td>
<td>Numeric, the upper bound of the conditional draw down.</td>
</tr>
<tr>
<td>softBudget</td>
<td>Logical, whether the budget constraint shall be implemented as a soft</td>
</tr>
<tr>
<td></td>
<td>constraint, i.e. the sum of the weights can be less than one. The default</td>
</tr>
<tr>
<td></td>
<td>is to use an equality constraint.</td>
</tr>
</tbody>
</table>

Arguments are passed down to Rglpk_solve_LP
Details

This function implements a long-only portfolio optimisation with a CDaR constraint (see references below). The problem can be stated in the form of a linear program and GLPK is used as solver.

Value

An object of formal class "PortAdd".

Note

A warning is issued in case the solver had exit status not equal to zero.

Author(s)

Bernhard Pfaff

References


See Also

"Portsol", "PortCdd", "PortDD", PMaxDD, PAveDD, PMinCDaR

Examples

```r
## Not run:
data(StockIndex)
popt <- PCDaR(PriceData = StockIndex, alpha = 0.95,
               bound = 0.1, softBudget = TRUE)

## End(Not run)
```

PERC

Equal risk contributed portfolios

Description

This function solves for equal risk contributed portfolio weights.

Usage

PERC(Sigma, par = NULL, percentage = TRUE, ...)
Arguments

Sigma Matrix, the variance-covariance matrix of asset returns
par Vector, the initial values of the weights.
percentage Logical, whether the weights shall be returned as decimals or percentages (default).
... Ellipsis argument is passed down to nlmnb().

Details

The objective function is the standard deviation of the marginal risk contributions, which is minimal, \textit{i.e.} zero, if all contributions are equal. The weights are rescaled to sum to unity.

Value

An object of formal class "PortSol".

Note

The optimisation is conducted by calling nlmnb(). Hereby, the arguments \texttt{lower} = 0 and \texttt{upper} = 1 have been specified.

Author(s)

Bernhard Pfaff

References


See Also

"PortSol", PERC2

Examples

data(MultiAsset)
Rets <- returnseries(MultiAsset, method = "discrete", trim = TRUE, percentage = TRUE)
V <- cov(Rets)
ERC <- PERC(V)
ERC
w <- Weights(ERC)
w * V
Description
This function solves for equal risk contributed portfolio weights by employing the solnp() function.

Usage
PERC2(Sigma, par = NULL, percentage = TRUE, ...)

Arguments
Sigma       Matrix, the variance-covariance matrix of asset returns
par        Vector, the initial values of the weights.
percentage Logical, whether the weights shall be returned as decimals or percentages (default).
... Ellipsis argument is passed down to solnp().

Details
The objective function is the standard deviation of the marginal risk contributions, which is minimal, i.e. zero, if all contributions are equal. The weights are rescaled to sum to unity. The lower and upper bounds are set to LB = rep(0, N) and UB = rep(1, N), respectively.

Value
An object of formal class "PortSol".

Note
The optimisation is conducted by calling solnp() contained in the package Rsolnp. Hereby, the arguments LB = 0 and UB = 1 have been specified.

Author(s)
Bernhard Pfaff

References

See Also
"PortSol", PERC
Examples

```r
## Not run:
library(Rsolnp)
data(MultiAsset)
Rets <- returnseries(MultiAsset, method = "discrete", trim = TRUE,
                 percentage = TRUE)
V <- cov(Rets)
## Budget Constraint
Budget <- function(x, Sigma) sum(x)
ERC <- PERC2(V, eqfun = Budget, eqB = 1)
ERC
w <- Weights(ERC) / 100
w * V
## End(Not run)
```

---

**PGMV**

*Global Minimum Variance Portfolio*

**Description**

This function returns the solution of the global minimum variance portfolio (long-only).

**Usage**

```r
PGMV(returns, percentage = TRUE, ...)
```

**Arguments**

- **returns**
  - A rectangular array of return data.
- **percentage**
  - Logical, whether the weights shall be returned as decimals or percentages (default).
- **...**
  - Arguments are passed down to cov.

**Value**

An object of formal class "PortSol".

**Note**

The optimisation is conducted by calling solve.QP().

**Author(s)**

Bernhard Pfaff

**See Also**

"PortSol"
Examples

```r
data(MultiAsset)
Ret = returnseries(MultiAsset, method = "discrete", trim = TRUE)
PGMV(Rets)
```

Description

Additional arguments to the `plot`-method pertinent to the defined S4-classes in this package are detailed below.

Usage

```r
## S4 method for signature 'PortDD'
plot(x, main = NULL, xlab = NULL, ylab = NULL,
col = c("black", "red"), grid = TRUE, invert = TRUE, ...)
```

Arguments

- `x` PortDD: an object that belongs to this virtual class.
- `main` character: The title of the plot.
- `xlab` character: The description of the x-axis.
- `ylab` character: The description of the y-axis.
- `col` character: Two-element vector of the names of the colors for the portfolio’s draw downs and the optimal level.
- `grid` Logical: Whether to superimpose a grid on the plot.
- `invert` Logical: Whether the draw downs shall be plotted as negative numbers; the default is TRUE.
- `...` Ellipsis argument is passed to the generic `plot` function.

Author(s)

Bernhard Pfaff
PMaxDD

Description
This function returns the result of a long-only portfolio optimization whereby the portfolio’s (historic) draw down is constrained to an upper limit.

Usage
PMaxDD(PriceData, MaxDD = 0.1, softBudget = FALSE, ...)

Arguments
- PriceData: A rectangular array of price data.
- MaxDD: Numeric, the upper bound of the maximum draw down.
- softBudget: Logical, whether the budget constraint shall be implemented as a soft constraint, i.e. the sum of the weights can be less than one. The default is to use an equality constraint.
- ...: Arguments are passed down to Rglpk_solve_LP

Details
This function implements a long-only portfolio optimisation with a maximum draw down constraint (see references below). The problem can be stated in the form of a linear program and GLPK is used as solver.

Value
An object of formal class "PortMdd".

Note
A warning is issued in case the solver had exit status not equal to zero.

Author(s)
Bernhard Pfaff

References
PMD

Most Diversified Portfolio

Description

This function returns the solution of the most diversified portfolio (long-only).

Usage

PMD(Returns, percentage = TRUE, ...)

Arguments

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>A rectangular array of return data.</td>
</tr>
<tr>
<td>percentage</td>
<td>Logical, whether the weights shall be returned as decimals or percentages</td>
</tr>
<tr>
<td>...</td>
<td>Arguments are passed down to cov().</td>
</tr>
</tbody>
</table>

Details

The optimisation problem is akin to that of a global minimum-variance portfolio, but instead of using the variance-covariance matrix of the asset returns, the correlation matrix is utilised as dispersion measure. The weights are then recovered by rescaling the optimal solution with the assets’ standard deviations and normalizing, such that the weights sum to one.

Value

An object of formal class "PortSol".

Note

The optimisation is conducted by calling solve.QP().

Author(s)

Bernhard Pfaff
References


See Also

"PortSol"

Examples

data(MultiAsset)
Rets <- returnseries(MultiAsset, method = "discrete", trim = TRUE)
PMD(Rets)

---

**PMinCDaR**

*Portfolio optimisation for minimum conditional draw down at risk*

Description

This function returns the result of a long-only portfolio optimization whereby the portfolio’s (historic) conditional draw down at risk is minimized.

Usage

`PMinCDaR(PriceData, alpha = 0.95, softBudget = FALSE, ...)`

Arguments

- **PriceData**: A rectangular array of price data.
- **alpha**: Numeric, the confidence level for which the conditional draw down shall be computed.
- **softBudget**: Logical, whether the budget constraint shall be implemented as a soft constraint, *i.e.* the sum of the weights can be less than one. The default is to use an equality constraint.
- **...**: Arguments are passed down to Rglpk_solve_LP

Details

This function implements a long-only portfolio optimisation for a minimum conditional draw down at risk (see references below). The problem can be stated in the form of a linear program and GLPK is used as solver.

Value

An object of formal class "PortAdd".
Note

A warning is issued in case the solver had exit status not equal to zero.

Author(s)

Bernhard Pfaff

References


See Also

"PortSol", "PortCdd", "PortDD", PMaxDD, PAveDD, PCDaR

Examples

```r
## Not run:
data(stockindex)
popt <- PMinCDaR(PriceData = stockindex, alpha = 0.95, softBudget = FALSE)

## End(Not run)
```

PMTD

Minimum Tail Dependent Portfolio

Description

This function computes the solution of a minimum tail dependent portfolio (long-only).

Usage

PMTD(returns, method = c("EmpTC", "EVT"), k = NULL, percentage = TRUE, ...)

Arguments

- **Returns**: A rectangular array of return data.
- **method**: Character, the type of non-parametric estimation.
- **k**: Integer, the threshold value for the order statistic. If left NULL, then \(k = \sqrt{nrow(x)}\) is used.
- **percentage**: Logical, whether the weights shall be returned as decimals or percentages (default).
- **...**: Arguments are passed down to `rank`. 
Details
Akin to the optimisation of a global minimum-variance portfolio, the minimum tail dependent portfolio is determined by replacing the variance-covariance matrix with the matrix of the lower tail dependence coefficients as returned by \texttt{tdc}.

Value
An object of formal class "\texttt{PortSol}".

Note
The optimisation is conducted by calling \texttt{solve.QP()}.

Author(s)
Bernhard Pfaff

See Also
\texttt{tdc}, "\texttt{PortSol}"

Examples
\begin{verbatim}
data(StockIndex)
Rets <- returnseries(StockIndex, method = "discrete", trim = TRUE,
percentage = TRUE)
PMTD(Rets)
\end{verbatim}

\begin{verbatim}
PortAdd-class          Class "PortAdd"
\end{verbatim}

Description
This class is intended to hold the results from a portfolio optimisation with a constraint on its average draw down.

Objects from the Class
Objects can be created by calls of the form \texttt{new("PortAdd", \ldots)}. This class extends the "\texttt{PortSol}" class.

Slots
\begin{verbatim}
AveDD: Numeric, the average draw down.
DrawDown: timeSeries, the historic portfolio’s draw downs.
weights: Numeric, vector of optimal weights.
opt: List, the result of the call to GLPK.
type: Character, the type of the optimized portfolio.
call: The call to the function that created the object.
\end{verbatim}
**PortCdd-class**

**Extends**

Class "PortSol", directly.

**Methods**

No methods defined with class "PortAdd" in the signature.

**Author(s)**

Bernhard Pfaff

**See Also**

"PortSol", "PortMdd", "PortCdd"

**Examples**

```
showClass("PortAdd")
```

---

**PortCdd-class**  
**Class "PortCdd"**

**Description**

This class is intended to hold the results from a portfolio optimisation with a constraint on its average draw down.

**Objects from the Class**

Objects can be created by calls of the form `new("PortCdd", ...`). This class extends the "PortSol" class.

**Slots**

- `cdar`: Numeric, the conditional draw down at risk.
- `thresh`: Numeric, threshold value for draw downs at the \(\alpha\) level.
- `DrawDown`: timeSeries, the historic portfolios draw downs.
- `weights`: Numeric, vector of optimal weights.
- `opt`: List, the result of the call to GLPK.
- `type`: Character, the type of the optimized portfolio.
- `call`: The call to the function that created the object.

**Extends**

Class "PortSol", directly.
Methods

No methods defined with class "PortCdd" in the signature.

Author(s)

Bernhard Pfaff

See Also

"PortSol", "PortMdd", "PortAdd"

Examples

showClass("PortCdd")
Description
This class is intended to hold the results from a portfolio optimisation with a constraint on its maximum draw down.

Objects from the Class
Objects can be created by calls of the form `new("PortMdd", ...). This class extends the "PortSol" class.

Slots
- `maxdd`: Numeric, the maximum draw down.
- `drawdown`: timeSeries, the historic portfolio’s draw downs.
- `weights`: Numeric, vector of optimal weights.
- `opt`: List, the result of the call to GLPK.
- `type`: Character, the type of the optimized portfolio.
- `call`: The call to the function that created the object.

Extends
Class "PortSol", directly.

Methods
No methods defined with class "PortMdd" in the signature.

Author(s)
Bernhard Pfaff

See Also
"PortSol", "PortAdd", "PortCdd"

Examples
`showClass("PortMdd")`
**PortSol-class**

**Class "PortSol"**

### Description

This class is intended to hold the results for the weights of an optimal portfolio. Currently, this class is used for minimum-variance and equal-risk-contributed portfolios. It can further be used to store the results of optimal factor weights according to one of the aforementioned portfolio types.

### Objects from the Class

Objects can be created by calls of the form `new("PortSol", ...)`.

### Slots

- `weights`: Numeric, vector of optimal weights.
- `opt`: List, the result of the call to the optimizing function.
- `type`: Character, the type of the optimized portfolio.
- `call`: The call to the function that created the object.

### Methods

- **show** signature(object = "PortSol"): Returns the portfolio type as text with the optimal weights from the object.
- **Solution** signature(object = "PortSol"): Returns the list object of the optimizer, *i.e.* the slot `opt` from the object.
- **Weights** signature(object = "PortSol"): Returns the list object of the optimizer, *i.e.* the slot `weights` from the object.
- **update** signature(object = "PortSol"): updates object by calling the issuing function with altered arguments.

### Author(s)

Bernhard Pfaff

### Examples

`showClass("PortSol")`
**returnconvert**  
*Convert Returns from continuous to discrete and vice versa*

**Description**
Either continuous returns or discrete returns can be converted into the other type.

**Usage**
```r
returnconvert(y, convdir = c("cont2disc", "disc2cont"), percentage = TRUE)
```

**Arguments**
- `y`: Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.
- `convdir`: Character, the type of return conversion.
- `percentage`: Logical, if TRUE (the default) the returns, `y`, are expressed as percentages.

**Value**
An object of the same class as `y`, containing the converted returns.

**Methods**
- `y = "data.frame"` The calculation is applied per column of the data.frame and only if all columns are numeric.
- `y = "matrix"` The calculation is applied per column of the matrix.
- `y = "mts"` The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
- `y = "numeric"` Calculation of the returns.
- `y = "timeSeries"` The calculation is applied per column of the timeSeries object and an object of the same class is returned.
- `y = "ts"` Calculation of the returns. The attributes are preserved and an object of the same class is returned.
- `y = "xts"` Calculation of the returns. The attributes are preserved and an object of the same class is returned.
- `y = "zoo"` Calculation of the returns. The attributes are preserved and an object of the same class is returned.

**Author(s)**
Bernhard Pfaff
Examples

data(StockIndex)
yc <- diff(log(StockIndex[, "SP500"])) * 100
yd <- returnseries(StockIndex[, "SP500"], method = "discrete",
percentage = TRUE, trim = TRUE)
yconv <- returnconvert(yd, convdir = "disc2cont",
percentage = TRUE)
all.equal(yc, yconv)

returnseries
Continuous and discrete returns

Description

Either continuous returns or discrete returns are computed for an object. The returns can be ex-
pressed as percents and the first NA value can be trimmed.

Usage

returnseries(y, method = c("continuous", "discrete"), percentage = TRUE,
trim = FALSE, compound = FALSE)

Arguments

y Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts
are supported.
method Character, the type of return to be computed.
percentage Logical, if TRUE (the default) the returns are expressed as percents.
trim Logical, if FALSE (the default) the first value is set to NA such that the length of
the return series coincides with the length of the series in levels.
compound Logical, if FALSE (the default), then simple returns are computed and otherwise
compounded returns.

Value

An object of the same class as y, containing the truncated series.

Methods

y = "data.frame" The calculation is applied per column of the data.frame and only if all columns
are numeric.
y = "matrix" The calculation is applied per column of the matrix.
y = "mts" The calculation is applied per column of the mts object. The attributes are preserved
and an object of the same class is returned.
y = "numeric" Calculation of the es trend.
y = "timeSeries"  The calculation is applied per column of the timeSeries object and an object of the same class is returned.

y = "ts"  Calculation of the returns. The attributes are preserved and an object of the same class is returned.

y = "xts"  Calculation of the returns. The attributes are preserved and an object of the same class is returned.

y = "zoo"  Calculation of the returns. The attributes are preserved and an object of the same class is returned.

Author(s)
Bernhard Pfaff

Examples

data(StockIndex)
y <- StockIndex[, "SP500"]
ret <- returnseries(y)
head(ret)

socp  Second-order Cone Programming

Description

The function solves second-order cone problem by primal-dual interior point method. It is a wrapper function to the C-routines written by Lobo, Vandenberghe and Boyd (see reference below).

Usage

socp(f, A, b, C, d, N,
    x = NULL, z = NULL, w = NULL, control = list())

Arguments

f        Vector defining linear objective, length(f)==length(x)
A        Matrix with the $A_i$ vertically stacked: $A = [A_1; A_2; \ldots; A_L]$.
b        Vector with the $b_i$ vertically stacked: $b = [b_1; b_2; \ldots; b_L]$.
C        Matrix with the $c_i'$ vertically stacked: $C = [c_1'; c_2'; \ldots; c_L']$.
d        Vector with the $d_i$ vertically stacked: $d = [d_1; d_2; \ldots; d_L]$.
N        Vector of size L, defining the size of each constraint.
x        Primal feasible initial point.  Must satisfy: $||A_i * x + b_i|| < c_i' * x + d_i$ for $i = 1, \ldots, L$.
z        Dual feasible initial point.
w        Dual feasible initial point.
control A list of control parameters.
Details

The primal formulation of an SOCP is given as:

\[
\text{minimise } f' \ast x
\]

subject to

\[
||A_i \ast x + b_i|| \leq c'_i \ast x + d_i
\]

for \( i = 1, \ldots, L \). Here, \( x \) is the \((n \times 1)\) vector to be optimised. The dual form of an SOCP is expressed as:

\[
\text{maximise } \sum_{i=1}^{L} -(b' \ast z_i + d_i \ast w_i)
\]

subject to

\[
\sum_{i=1}^{L} (A'_i \ast z_i + c_i \ast w_i) = f
\]

and

\[
||z_i|| = w_i
\]

for \( i = 1, \ldots, L \), given strictly feasible primal and dual initial points.

The algorithm stops, if one of the following criteria is met:

1. \text{abs.tol} – maximum absolute error in objective function; guarantees that for any \( x \): \( \text{abs}(f' \ast x - f' \ast x_{\text{opt}}) \leq \text{abs.tol} \).
2. \text{rel.tol} – maximum relative error in objective function; guarantees that for any \( x \): \( \text{abs}(f' \ast x - f' \ast x_{\text{opt}})/(f' \ast x_{\text{opt}}) \leq \text{rel.tol}(\text{if } f' \ast x_{\text{opt}} > 0) \). Negative value has special meaning, see target next.
3. \text{target} – if \( \text{rel.tol} < 0 \), stops when \( f' \ast x < \text{target} \) or \( -b' \ast z > \text{target} \).
4. \text{max.iter} – limit on number of algorithm outer iterations. Most problems can be solved in less than 50 iterations. Called with \( \text{max.iter} = 0 \) only checks feasibility of \( x \) and \( z \), (and returns gap and deviation from centrality).
5. The target value is reached. \( \text{rel.tol} \) is negative and the primal objective \( p \) is less than the target.

Value

A list-object with the following elements:

- \( x \) Solution to the primal problem.
- \( z \) Solution to the dual problem.
- \( \text{iter} \) Number of iterations performed.
- \( \text{hist} \) see \text{out.mode} in \text{SocpControl}.
- \( \text{convergence} \) A logical code. \text{TRUE} indicates successful convergence.
- \( \text{info} \) A numerical code. It indicates if the convergence was successful.
- \( \text{message} \) A character string giving any additional information returned by the optimiser.
Note

This function has been ported from the Rsocp package contained in the Rmetrics-Project on R-Forge. In contrast to the former implementation, allowance is made for specifying more than one cone constraint.

Author(s)

Bernhard Pfaff

References


See Also

SocpPhase1, SocpPhase2, SocpControl

---

**SocpControl**  
Control Variables for Socp

---

**Description**

This function returns a list object of control parameters that are passed down to the C-function SOCP. Its default values are used in Socp.

**Usage**

```r
SocpControl(abs.tol = 1e-18, rel.tol = 1e-16, target = 0,
             max.iter = 500, Nu = 10, out.mode = 0, BigM.K = 2,
             BigM.iter = 5)
```

**Arguments**

- `abs.tol`  
  Absolute tolerance.
- `rel.tol`  
  Relative tolerance.
- `target`  
  Target value < 0, only used if rel.tol < 0.
- `max.iter`  
  The maximum number of iterations, socp is aborted if more are required for convergence.
- `Nu`  
  The parameter that controls the rate of convergence, Nu > 1, recommended range 5 to 50.
- `out.mode`  
  Specifies what should be output: 0 - nothing, 1 - duality gap for initial point and after each iteration, 2 - duality gap and deviation from centrality, for initial point and after each iteration.
- `BigM.K`  
  Iteration parameter. The default values is `BigM.K = 2`.
- `BigM.iter`  
  Iteration parameter. The default values is `BigM.iter = 5`.  

Details

For details about these control parameters, the reader is referred to the reference below, in particular sections 2.7, 2.8 and 4.3 to 4.5. A pdf-version of the user’s guide is shipped in the packages doc subdirectory.

Value

A list object with the control parameters.

Note

This function has been ported from the Rsocp package contained in the Rmetrics-Project on R-Forge.

Author(s)

Bernhard Pfaff

References


See Also

Socp

**Description**

This function determines values for \( x \), whence they have not been specified by the user. Here, a feasibility problem is solved first and its solution is then used as an initial point for the original problem.

**Usage**

```r
SocpPhase1(f, A, b, N, control)
```

**Arguments**

- `f` vector: the parameters of the objective function in its primal form.
- `A` matrix: the parameter matrix of the cone constraints.
- `b` vector: the parameter vector of the cone constraints.
- `N` vector: the count of rows pertinent to each cone constraint.
- `control` list: the list of control parameters for SOCP.
Details
The finding of an initial point \( x \) is described in the user's guide, section 2.8.

Value
A vector with the initial point for \( x \).

Note
This function has been ported from the \texttt{Rsocp} package contained in the Rmetrics-Project on R-Forge.

Author(s)
Bernhard Pfaff

References

See Also
\texttt{Scop}, \texttt{ScopPhase2}, \texttt{ScopControl}

---

\texttt{ScopPhase2} \hspace{1cm} \textit{SOCP: Initialising objective variable \( z \) in dual form}

Description
This function determines values for \( z \), whence they have not been specified by the user.

Usage
\texttt{ScopPhase2(f, A, b, N, x, control)}

Arguments
- \( f \) \hspace{1cm} \text{vector: the parameters of the objective function in its primal form.}
- \( A \) \hspace{1cm} \text{matrix: the parameter matrix of the cone constraints.}
- \( b \) \hspace{1cm} \text{vector: the parameter vector of the cone constraints.}
- \( N \) \hspace{1cm} \text{vector: the count of rows pertinent to each cone constraint.}
- \( x \) \hspace{1cm} \text{vector: initial point of SOCP in its primal form.}
- \texttt{control} \hspace{1cm} \text{list: the list of control parameters for SOCP.}
Value

A vector with the initial point for \( z \) (dual form of SOCP).

Note

This function has been ported from the \texttt{Rsocp} package contained in the Rmetrics-Project on R-Forge.

Author(s)

Bernhard Pfaff

References


See Also

\texttt{socp, socpPhase1, socpControl}

---

\textbf{SP500} \hspace{2cm} \textit{Standard & Poor’s 500}

**Description**


**Usage**

data(SP500)

**Format**

A data frame with 265 weekly observations of 476 members of the S&P 500 index. The sample starts at 2003-03-03 and ends in 2008-03-24.

**Details**

The data set was used in the reference below. The authors adjusted the price data for dividends and have removed stocks if two or more consecutive missing values were found. In the remaining cases the NA entries have been replaced by interpolated values.

**Source**

http://w3.uniroma1.it/Tardella/datasets.html

http://finance.yahoo.com/
**sqrm**

**References**


**Examples**

```r
data(SP500)
```

---

**sqrm**  
*Square root of a quadratic matrix*

**Description**

This function returns the square root of a quadratic and diagonalisable matrix.

**Usage**

```r
sqrm(x, ...)
```

**Arguments**

- `x`  
  matrix, must be quadratic.

- `...`  
  The ellipsis argument is passed down to `eigen()`.

**Details**

The computation of the square root of a matrix is based upon its eigen values and corresponding eigen vectors. The square matrix $A$ is diagonalisable if there is a matrix $V$ such that $D = V^{-1} A V$, whereby $D$ is a diagonal matrix. This is only achieved if the eigen vectors of the $(n \times n)$ matrix $A$ constitute a basis of dimension $n$. The square root of $A$ is then $A^{1/2} = V D^{1/2} V^\prime$.

**Value**

A matrix object and a scalar in case a $(1 \times 1)$ matrix has been provided.

**Author(s)**

Bernhard Pfaff

**See Also**

`eigen`
Examples

```r
data(StockIndex)
S <- cov(StockIndex)
SR <- sqrm(S)
all.equal(crossprod(SR), S)
```

---

### StockIndex

**Stock Index Data**

**Description**

Month-end price data of six stock indices.

**Usage**

```r
data(StockIndex)
```

**Format**


**Details**

The data set has been obtained from Yahoo Finance and hereby the unadjusted closing prices have been retrieved.

**Source**


**Examples**

```r
data(StockIndex)
```
**Description**

Adjusted month-end price data of six stock indices.

**Usage**

```r
data(StockIndexAdj)
```

**Format**


**Details**

The data set has been obtained from Yahoo Finance and hereby the adjusted closing prices have been retrieved.

**Source**


**Examples**

```r
data(StockIndexAdj)
```

---

**Description**

Adjusted daily price data of six stock indices.

**Usage**

```r
data(StockIndexAdjD)
```

**Format**

Details

The data set has been obtained from Yahoo Finance and hereby the adjusted closing prices have been retrieved.

Source

http://finance.yahoo.com/

Examples

data(StockIndexAdj)

tdc

Tail Dependence Coefficient

Description

This function returns the pairwise tail dependence coefficients between \( N \) series. The TDCs are estimated non-parametrically by either the empirical tail copula or based on the stable tail-dependence function.

Usage

tdc(x, method = c("EmpTC", "EVT"), lower = TRUE, k = NULL, ...)

Arguments

- \( x \): Matrix, or an object that can be coerced to it.
- \( \text{method} \): Character, the type of non-parametric estimation.
- \( \text{lower} \): Logical, if \( \text{TRUE} \) (default), lower TDC are computed and upper TDC, else.
- \( k \): Integer, the threshold value for the order statistic. If left \( \text{NULL} \), then \( k = \sqrt{nrow(x)} \) is used.
- \( \ldots \): Ellipsis, arguments are passed down to \( \text{rank} \).

Details

For a matrix or an object that can be coerced to it with \( ncol(x) \geq 2 \), the pairwise tail dependencies are estimated non-parametrically and returned as a symmetric matrix. The threshold value \( k \) is the upper/lower bound for the order statistics to be considered. The diagonal elements are always equal to one, because a series has a dependence of one with itself, of course.

Value

A matrix with the tail dependent coefficients.
Author(s)

Bernhard Pfaff

References


See Also

PMTD

Examples

data(StockIndex)
Rets <- returnseries(StockIndex, method = "discrete", trim = TRUE, percentage = TRUE)
tdc(Rets, method = "EmpTC")
tdc(Rets, method = "EVT")

**trdbilson**  
*Bilson Trend*

Description

Calculation of the Bilson Trend as a technical trading indicator.

Usage

`trdbilson(y, exponent)`

Arguments

- `y`  
  Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.

- `exponent`  
  Numeric, the value for $\alpha$ in the equation below.

Details

The Bilson trend is calculated according to the formula:

$$z = \text{sign}(y) \times |y|^{(1-|y|^\alpha)}$$

Value

An object of the same class as `y`, containing the computed Bilson trend values.
Methods

- **y = "data.frame"** The calculation is applied per column of the data.frame and only if all columns are numeric.
- **y = "matrix"** The calculation is applied per column of the matrix.
- **y = "mts"** The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
- **y = "numeric"** Calculation of the bilson trend.
- **y = "timeSeries"** The calculation is applied per column of the timeSeries object and an object of the same class is returned.
- **y = "ts"** Calculation of the bilson trend. The attributes are preserved and an object of the same class is returned.
- **y = "xts"** Calculation of the bilson trend. The attributes are preserved and an object of the same class is returned.
- **y = "zoo"** Calculation of the bilson trend. The attributes are preserved and an object of the same class is returned.

Author(s)

Bernhard Pfaff

See Also

- `trdbinary`, `trdes`, `trdhp`, `trdsma`, `trdwma`, `capser`

Examples

```r
data(StockIndex)
y <- StockIndex[, "SP500"]
yret <- diff(log(y))
bilson <- trdbilson(yret, exponent = 2)
head(bilson)
```

---

**trdbinary**  

*Binary Trend*

Description

Calculation of the Binary Trend as a technical trading indicator.

Usage

`trdbinary(y)`
Arguments

y Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.

Details

The Binary trend is calculated according to the formula:

\[ z = \text{sign}(y) \times \min\left(|\frac{4}{\pi} \arctan(y)|, 1\right) \]

Value

An object of the same class as y, containing the computed Binary trend values.

Methods

y = "data.frame" The calculation is applied per column of the data.frame and only if all columns are numeric.

y = "matrix" The calculation is applied per column of the matrix.

y = "mts" The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.

y = "numeric" Calculation of the binary trend.

y = "timeSeries" The calculation is applied per column of the timeSeries object and an object of the same class is returned.

y = "ts" Calculation of the binary trend. The attributes are preserved and an object of the same class is returned.

y = "xts" Calculation of the binary trend. The attributes are preserved and an object of the same class is returned.

y = "zoo" Calculation of the binary trend. The attributes are preserved and an object of the same class is returned.

Author(s)

Bernhard Pfaff

See Also

trdbilson, trdes, trdhp, trdsma, trdwma, capser

Examples

data(StockIndex)
y <- StockIndex[, "SP500"]
yret <- diff(log(y))
binary <- trdbinary(yret)
head(binary)
Exponentially Smoothed Trend

Description
Calculation of the exponentially smoothed trend as a technical trading indicator.

Usage
trdes(y, lambda, init = NULL)

Arguments
- **y**: Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.
- **lambda**: Numeric, the smoothing parameter for \( \lambda \) in the equation below. The value for the parameter must be in the interval \( 0 < \lambda < 1 \).
- **init**: The initial value in the recursive calculation of the filter. Specifies the initial values of the time series just prior to the start value, in reverse time order. The default, *i.e.* NULL, is a set of zeros.

Details
The exponentially smoothed trend is calculated according to the formula:

\[
z_t = \lambda y_t + (1 - \lambda) * z_{t-1}
\]

Value
An object of the same class as \( y \), containing the computed exponentially smoothed values.

Methods
- **y = "data.frame"**: The calculation is applied per column of the data.frame and only if all columns are numeric.
- **y = "matrix"**: The calculation is applied per column of the matrix.
- **y = "mts"**: The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
- **y = "numeric"**: Calculation of the es trend.
- **y = "timeSeries"**: The calculation is applied per column of the timeSeries object and an object of the same class is returned.
- **y = "ts"**: Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
- **y = "xts"**: Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
- **y = "zoo"**: Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
**trdhp**

**Author(s)**
Bernhard Pfaff

**See Also**
`filter, trdbilson, trdbinary, trdhp, trdsma, trdwma, capser`

**Examples**
```r
data(StockIndex)
y <- StockIndex[, "SP500"]
yret <- diff(log(y))
es <- trdes(yret, lambda = 0.95)
head(es)
```

---

**Description**
Calculation of the Hodrick-Prescott filter as a technical trading indicator.

**Usage**

```
trdhp(y, lambda)
```

**Arguments**
- `y` Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.
- `lambda` Numeric, the value for \( \lambda \) in the equation below.

**Details**
The Hodrick-Prescott filter is calculated according to the formula:

\[
\min(\tau_t) = \sum_{t=1}^{T}(y_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1}(\Delta^2 \tau_{t+1})^2
\]

**Value**
An object of the same class as \( y \), containing the computed Hodrick-Prescott values.
Methods

- **y = "data.frame"** The calculation is applied per column of the data.frame and only if all columns are numeric.
- **y = "matrix"** The calculation is applied per column of the matrix.
- **y = "mts"** The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
- **y = "numeric"** Calculation of the bilson trend.
- **y = "timeSeries"** The calculation is applied per column of the timeSeries object and an object of the same class is returned.
- **y = "ts"** Calculation of the bilson trend. The attributes are preserved and an object of the same class is returned.
- **y = "xts"** Calculation of the bilson trend. The attributes are preserved and an object of the same class is returned.
- **y = "zoo"** Calculation of the bilson trend. The attributes are preserved and an object of the same class is returned.

Author(s)

Bernhard Pfaff

References


See Also

`trdbinary, trdes, trdbilson, trdsma, trdwma, capser`

Examples

```r
data(StockIndex)
y <- StockIndex[, "SP500"]
hp <- trdhp(y, lambda = 1600)
head(hp)
```

---

### trdsma

**Simple Moving Average**

Calculation of a right ended simple moving average with equal weights determined by `n.periods`.

**Usage**

```
trdsma(y, n.periods, trim = FALSE)
```
Arguments

- `y` Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.
- `n.periods` Integer, the number of periods to be included in the calculation of the simple moving average.
- `trim` Logical, if FALSE (the default) the first value is set to NA, otherwise the object is trimmed by the first observation.

Value

An object of the same class as `y`, containing the computed simple moving averages.

Methods

- `y = "data.frame"` The calculation is applied per column of the data.frame and only if all columns are numeric.
- `y = "matrix"` The calculation is applied per column of the matrix.
- `y = "mts"` The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
- `y = "numeric"` Calculation of the es trend.
- `y = "timeSeries"` The calculation is applied per column of the timeSeries object and an object of the same class is returned.
- `y = "ts"` Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
- `y = "xts"` Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
- `y = "zoo"` Calculation of the es trend. The attributes are preserved and an object of the same class is returned.

Author(s)

Bernhard Pfaff

See Also

`filter, trdbilson, trdbinary, trdhp, trdwm, capser, trdes`

Examples

```r
data(StockIndex)
y <- StockIndex[, "SP500"]
sma <- trdsma(y, n.periods = 24)
head(sma, 30)
```
trdwma

Weighted Moving Average

Description
Calculation of a right ended weighted moving average with weights according to weights.

Usage
trdwma(y, weights, trim = FALSE)

Arguments
y  Objects of classes: numeric, matrix, data.frame, ts, mts, timeSeries, zoo and xts are supported.
weights Numeric, a vector containing the weights.
trim Logical, if FALSE (the default) the first value is set to NA, otherwise the object is trimmed by the first observation.

Details
If the sum of the weights is greater than unity, a warning is issued.

Value
An object of the same class as y, containing the computed weighted moving averages.

Methods
y = "data.frame" The calculation is applied per column of the data.frame and only if all columns are numeric.
y = "matrix" The calculation is applied per column of the matrix.
y = "mts" The calculation is applied per column of the mts object. The attributes are preserved and an object of the same class is returned.
y = "numeric" Calculation of the es trend.
y = "timeSeries" The calculation is applied per column of the timeSeries object and an object of the same class is returned.
y = "ts" Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
y = "xts" Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
y = "zoo" Calculation of the es trend. The attributes are preserved and an object of the same class is returned.
trdwma

Author(s)

Bernhard Pfaff

See Also

filter, trdbilson, trdbinary, trdhp, trdes, trdsma, capser

Examples

data(StockIndex)
y <- StockIndex[, "SP500"]
wma <- trdwma(y, weights = c(0.4, 0.3, 0.2, 0.1))
head(wma, 30)
Index

*Topic **IO**
BookEx, 3

*Topic **algebra**
DivMeasures, 5
mrc, 17
sqrn, 43
tdc, 46

*Topic **array**
sqrn, 43

*Topic **classes**
PortAdd-class, 30
PortCdd-class, 31
PortDD-class, 32
PortMdd-class, 33
PortSol-class, 34

*Topic **datasets**
ESCBFX, 7
EuroStoxx50, 8
FTSE100, 9
INDTRACK1, 10
INDTRACK2, 11
INDTRACK3, 12
INDTRACK4, 13
INDTRACK5, 14
INDTRACK6, 15
MITTEL, 16
MultiAsset, 17
NASDAQ, 18
SP500, 42
StockIndex, 44
StockIndexAdj, 45
StockIndexAdjD, 45

*Topic **methods**
capser, 4
plot-methods, 25
returnconvert, 35
returnsseries, 36
trdbilson, 47
trdbinary, 48
trdes, 50
trdhpc, 51
trdsma, 52
trdwma, 54

*Topic **optimize**
PAveDD, 19
PCDaR, 20
PERC, 21
PERC2, 23
PGMV, 24
PMAXDD, 26
PMD, 27
PMInCDaR, 28
PMTD, 29
Socop, 37
SocopControl, 39
SocopPhase1, 40
SocopPhase2, 41

*Topic **smooth**
capser, 4
trdbilson, 47
trdbinary, 48
trdes, 50
trdhpc, 51
trdsma, 52
trdwma, 54

*Topic **ts**
capser, 4
returnconvert, 35
returnsseries, 36
trdbilson, 47
trdbinary, 48
trdes, 50
trdhpc, 51
trdsma, 52
trdwma, 54

BookEx, 3

capser, 4, 48, 49, 51–53, 55
INDEX

capser, data.frame-method (capser), 4
capser, matrix-method (capser), 4
capser, mts-method (capser), 4
capser, numeric-method (capser), 4
capser, timeSeries-method (capser), 4
capser, ts-method (capser), 4
capser, xts-method (capser), 4
capser, zoo-method (capser), 4
capser-methods (capser), 4
cr (DivMeasures), 5
DivMeasures, 5
dr (DivMeasures), 5
DrawDowns (PortDD-class), 32
DrawDowns, PortDD-method (PortDD-class), 32

editEx (BookEx), 3
eigen, 43
ESCBFX, 7
EuroStoxx50, 8

file.edit, 4
filter, 51, 53, 55
FTSE100, 9

INDTRACK1, 10
INDTRACK2, 11
INDTRACK3, 12
INDTRACK4, 13
INDTRACK5, 14
INDTRACK6, 15

listEx (BookEx), 3

MIBTEL, 16
mrc, 17
MultiAsset, 17

NASDAQ, 18

PAvDD, 19, 21, 27, 29, 32
PCdAR, 20, 21, 27, 29, 32
PERC, 21, 23
PERC2, 22, 23
PGMV, 24
plot (PortDD-class), 32
plot, PortDD, missing-method (PortDD-class), 32
plot, PortDD-method (plot-methods), 25
plot-methods, 25
PMaxDD, 20, 21, 26, 29, 32
PMD, 6, 27
PMinCDaR, 20, 21, 27, 28, 32
PMD, 29, 47
PortAdd, 20, 32, 33
PortAdd-class, 30
PortCdd, 21, 29, 31–33
PortCdd-class, 31
PortDD, 20, 21, 27, 29
PortDD-class, 32
PortMdd, 27, 31, 32
PortMdd-class, 33
PortSol, 20–24, 27–33
PortSol-class, 34

returnconvert, 35
returnconvert, data.frame-method (returnconvert), 35
returnconvert, matrix-method (returnconvert), 35
returnconvert, mts-method (returnconvert), 35
returnconvert, numeric-method (returnconvert), 35
returnconvert, timeSeries-method (returnconvert), 35
returnconvert, ts-method (returnconvert), 35
returnconvert, xts-method (returnconvert), 35
returnconvert, zoo-method (returnconvert), 35
returnconvert-methods (returnconvert), 35

returnseries, 36
returnseries, data.frame-method (returnseries), 36
returnseries, matrix-method (returnseries), 36
returnseries, mts-method (returnseries), 36
returnseries, numeric-method (returnseries), 36
returnseries, timeSeries-method (returnseries), 36
returnseries, ts-method (returnseries), 36
returnseries, xts-method (returnseries), 36
returnseries, zoo-method (returnseries), 36
returnseries-methods (returnseries), 36
rhow (DivMeasures), 5
runEx (BookEx), 3
saveEx (BookEx), 3
show, PortSol-method (PortSol-class), 34
showEx (BookEx), 3
Socp, 37, 40–42
SocpControl, 38, 39, 41, 42
SocpPhase1, 39, 40, 42
SocpPhase2, 39, 41, 41
Solution (PortSol-class), 34
Solution, PortSol-method (PortSol-class), 34
source, 4
SP500, 42
sqrm, 43
StockIndex, 44
StockIndexAdj, 45
StockIndexAdjD, 45
tdc, 30, 46
trdbilson, 5, 47, 49, 51–53, 55
trdbilson, data.frame-method (trdbilson), 47
trdbilson, matrix-method (trdbilson), 47
trdbilson, mts-method (trdbilson), 47
trdbilson, numeric-method (trdbilson), 47
trdbilson, timeSeries-method (trdbilson), 47
trdbilson, ts-method (trdbilson), 47
trdbilson, xts-method (trdbilson), 47
trdbilson, zoo-method (trdbilson), 47
trdbilson-methods (trdbilson), 47
trdbinary, 5, 48, 49, 51–53, 55
trdbinary, data.frame-method (trdbinary), 48
trdbinary, matrix-method (trdbinary), 48
trdbinary, mts-method (trdbinary), 48
trdbinary, numeric-method (trdbinary), 48
trdbinary, timeSeries-method (trdbinary), 48
trdbinary, ts-method (trdbinary), 48
trdbinary, xts-method (trdbinary), 48
trdbinary, zoo-method (trdbinary), 48
trdbinary-methods (trdbinary), 48
trdes, 5, 48, 49, 50, 52, 53, 55
trdes, data.frame-method (trdes), 50
trdes, matrix-method (trdes), 50
trdes, mts-method (trdes), 50
trdes, numeric-method (trdes), 50
trdes, timeSeries-method (trdes), 50
trdes, ts-method (trdes), 50
trdes, xts-method (trdes), 50
trdes, zoo-method (trdes), 50
trdes-methods (trdes), 50
trdhp, 5, 48, 49, 51, 53, 55
trdhp, data.frame-method (trdhp), 51
trdhp, matrix-method (trdhp), 51
trdhp, mts-method (trdhp), 51
trdhp, numeric-method (trdhp), 51
trdhp, timeSeries-method (trdhp), 51
trdhp, ts-method (trdhp), 51
trdhp, xts-method (trdhp), 51
trdhp, zoo-method (trdhp), 51
trdhp-methods (trdhp), 51
trdmsa, 5, 48, 49, 51, 52, 52, 55
trdmsa, data.frame-method (trdmsa), 52
trdmsa, matrix-method (trdmsa), 52
trdmsa, mts-method (trdmsa), 52
trdmsa, numeric-method (trdmsa), 52
trdmsa, timeSeries-method (trdmsa), 52
trdmsa, ts-method (trdmsa), 52
trdmsa, xts-method (trdmsa), 52
trdmsa, zoo-method (trdmsa), 52
trdmsa-methods (trdmsa), 52
trdwma, 5, 48, 49, 51–53, 54
trdwma, data.frame-method (trdwma), 54
trdwma, matrix-method (trdwma), 54
trdwma, mts-method (trdwma), 54
trdwma, numeric-method (trdwma), 54
trdwma, timeSeries-method (trdwma), 54
trdwma, ts-method (trdwma), 54
trdwma, xts-method (trdwma), 54
trdwma, zoo-method (trdwma), 54
trdwma-methods (trdwma), 54
update, PortSol-method (PortSol-class), 34
Weights (PortSol-class), 34
Weights, PortSol-method (PortSol-class), 34