Package ‘fPortfolio’

February 19, 2015

Title  Rmetrics - Portfolio Selection and Optimization
Date   2014-10-30
Version 3011.81
Author  Rmetrics Core Team,
        Diethelm Wuertz [aut],
        Tobias Setz [cre],
        Yohan Chalabi [ctb]
Maintainer Tobias Setz <tobias.setz@rmetrics.org>
Description Environment for teaching
        "Financial Engineering and Computational Finance".
Depends  R (>= 2.15.1), methods, timeDate, timeSeries, fBasics, fAssets
Imports fCopulae, robustbase, MASS, Rglpk, slam, Rsymphony, Rsolnp,
        kernlab, quadprog, rneos
Suggests Rsocp, Rnlminb2, Rdonlp2, dplR, bcp, fGarch, mvoutlier
Additional_repositories http://r-forge.r-project.org/
Note  SEVERAL PARTS ARE STILL PRELIMINARY AND MAY BE CHANGED IN THE
        FUTURE. THIS TYPICALLY INCLUDES FUNCTION AND ARGUMENT NAMES, AS
        WELL AS DEFAULTS FOR ARGUMENTS AND RETURN VALUES.
LazyData yes
License GPL (>= 2)
URL  https://www.rmetrics.org
NeedsCompilation no
Repository CRAN
Date/Publication 2014-10-30 15:54:08

R topics documented:

  fPortfolio-package .................................................. 3
  backtest-constructors .......................................... 3
  backtest-extractors ........................................... 4
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>backtest-functions</td>
</tr>
<tr>
<td>backtest-getMethods</td>
</tr>
<tr>
<td>backtest-performance</td>
</tr>
<tr>
<td>backtest-plots</td>
</tr>
<tr>
<td>backtest-portfolio</td>
</tr>
<tr>
<td>backtest-specification</td>
</tr>
<tr>
<td>backtestStats</td>
</tr>
<tr>
<td>data-sets</td>
</tr>
<tr>
<td>fPFOLIOBACKTEST</td>
</tr>
<tr>
<td>fPFOLIOCON</td>
</tr>
<tr>
<td>fPFOLIODATA</td>
</tr>
<tr>
<td>fPFOLIOSPEC</td>
</tr>
<tr>
<td>fPFOLIOVAL</td>
</tr>
<tr>
<td>fPORTFOLIO</td>
</tr>
<tr>
<td>frontier-plot</td>
</tr>
<tr>
<td>frontier-plotControl</td>
</tr>
<tr>
<td>frontier-points</td>
</tr>
<tr>
<td>mathprog-LP</td>
</tr>
<tr>
<td>mathprog-NLP</td>
</tr>
<tr>
<td>mathprog-QP</td>
</tr>
<tr>
<td>methods-plot</td>
</tr>
<tr>
<td>methods-show</td>
</tr>
<tr>
<td>methods-summary</td>
</tr>
<tr>
<td>monitor-stability</td>
</tr>
<tr>
<td>portfolio-constraints</td>
</tr>
<tr>
<td>portfolio-covEstimator</td>
</tr>
<tr>
<td>portfolio-dataSets</td>
</tr>
<tr>
<td>portfolio-efficientPortfolio</td>
</tr>
<tr>
<td>portfolio-feasiblePortfolio</td>
</tr>
<tr>
<td>portfolio-getData</td>
</tr>
<tr>
<td>portfolio-getDefault</td>
</tr>
<tr>
<td>portfolio-getPortfolio</td>
</tr>
<tr>
<td>portfolio-getSpec</td>
</tr>
<tr>
<td>portfolio-getVal</td>
</tr>
<tr>
<td>portfolio-pfolioRisk</td>
</tr>
<tr>
<td>portfolio-portfolioFrontier</td>
</tr>
<tr>
<td>portfolio-portfolioSpec</td>
</tr>
<tr>
<td>portfolio-riskPfolio</td>
</tr>
<tr>
<td>portfolio-rollingPortfolios</td>
</tr>
<tr>
<td>portfolio-setSpec</td>
</tr>
<tr>
<td>risk-budgeting</td>
</tr>
<tr>
<td>risk-surfaceRisk</td>
</tr>
<tr>
<td>risk-ternaryMap</td>
</tr>
<tr>
<td>solve-environment</td>
</tr>
<tr>
<td>solver-ampl</td>
</tr>
<tr>
<td>solver-family</td>
</tr>
<tr>
<td>utils-methods</td>
</tr>
<tr>
<td>weights-linePlot</td>
</tr>
</tbody>
</table>
Description

The Rmetrics “fPortfolio” package is a very powerful collection of functions to optimize portfolios and to analyze them from different points of view.

Details

Package: fPortfolio
Type: Package
Date: 2011
License: GPL Version 2 or later
Copyright: (c) 1999-2011 Diethelm Wuertz and Rmetrics Association
URL: http://www.rmetrics.org

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Description

Functions to set specifications for portfolio backtesting.

The functions are:

- `setWindowsFun`: Sets Windows function,
- `setWindowsParams`: Sets additional parameters for rolling windows function,
- `setWindowsHorizon`: Sets Windows horizon,
- `setStrategyFun`: Sets the portfolio Strategy function,
- `setStrategyParams`: Sets additional parameters for Strategy function,
- `setSmotherFun`: Sets the Smoother function,
backtest-extractors

setSmootherParams backtest | Sets additional parameters for Smoother function,
setSmootherLambda backtest | Sets the smoothing parameter Lambda,
setSmootherDoubleSmoothing backtest | Sets setting for double smoothing,
setSmootherInitialWeights backtest | Sets the initial weights to used in the smoothing,
setSmootherSkip backtest | Sets the number of skipped months.

Usage

setWindowsFun(backtest) <- value
setWindowsParams(backtest) <- value
setWindowsHorizon(backtest) <- value

setStrategyFun(backtest) <- value
setStrategyParams(backtest) <- value

setSmootherFun(backtest) <- value
setSmootherParams(backtest) <- value
setSmootherLambda(backtest) <- value
setSmootherDoubleSmoothing(backtest) <- value
setSmootherInitialWeights(backtest) <- value
setSmootherSkip(backtest) <- value

Arguments

backtest an S4 object of class fPFOLIOBACKTEST, the specification to be modified, by default the default of the function portfolioBacktest().
value a value for that component of backtest to be set. Note for setting Params value is a list.

Details

The function portfolioBacktest() allows to set the values for the specification structure from scratch.
To modify individual settings one can use the set functions.

References

Würtz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

backtest-extractors Portfolio backtest specification extractors

Description

Extracts information from an object of class fPFOLIOBACKTEST.

The functions are:
**backtest-extractors**  

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getWindows</td>
<td>Extract windows slot,</td>
</tr>
<tr>
<td>getWindowsFun</td>
<td>extract windows function,</td>
</tr>
<tr>
<td>getWindowsParams</td>
<td>extract a list of windows specific parameters,</td>
</tr>
<tr>
<td>getWindowsHorizon</td>
<td>extract windows horizon,</td>
</tr>
<tr>
<td>getStrategy</td>
<td>extract strategy slot,</td>
</tr>
<tr>
<td>getStrategyFun</td>
<td>extract the portfolio strategy function,</td>
</tr>
<tr>
<td>getStrategyParams</td>
<td>extract a list of portfolio strategy specific parameters,</td>
</tr>
<tr>
<td>getSmooter</td>
<td>extract the smoother slot,</td>
</tr>
<tr>
<td>getSmooterFun</td>
<td>Extract the Smoother function,</td>
</tr>
<tr>
<td>getSmooterParams</td>
<td>extract a list of Smoothing specific parameters,</td>
</tr>
<tr>
<td>getSmooterLambda</td>
<td>extract the smoothing parameter Lambda,</td>
</tr>
<tr>
<td>getSmotherDoubleSmoothing</td>
<td>extract setting for double smoothing,</td>
</tr>
<tr>
<td>getSmotherInitialWeights</td>
<td>extract the initial weights to used in the smoothing,</td>
</tr>
<tr>
<td>getSmootherskip</td>
<td>extract the number of skipped months,</td>
</tr>
<tr>
<td>getMessages</td>
<td>extract the message slot.</td>
</tr>
</tbody>
</table>

**Usage**

```r
## S3 method for class 'FPFOLIOBACKTEST'
getWindows(object)
## S3 method for class 'FPFOLIOBACKTEST'
getWindowsFun(object)
## S3 method for class 'FPFOLIOBACKTEST'
getWindowsParams(object)
## S3 method for class 'FPFOLIOBACKTEST'
getWindowsHorizon(object)

## S3 method for class 'FPFOLIOBACKTEST'
getStrategy(object)
## S3 method for class 'FPFOLIOBACKTEST'
getStrategyFun(object)
## S3 method for class 'FPFOLIOBACKTEST'
getStrategyParams(object)

## S3 method for class 'FPFOLIOBACKTEST'
getSmooter(object)
## S3 method for class 'FPFOLIOBACKTEST'
getSmooterFun(object)
## S3 method for class 'FPFOLIOBACKTEST'
getSmooterParams(object)
## S3 method for class 'FPFOLIOBACKTEST'
getSmooterLambda(object)
## S3 method for class 'FPFOLIOBACKTEST'
getSmotherDoubleSmoothing(object)
## S3 method for class 'FPFOLIOBACKTEST'
getSmotherInitialWeights(object)
## S3 method for class 'FPFOLIOBACKTEST'
getSmootherskip(object)
```
## Backtest-Functions

### User defined functions to perform portfolio backtesting

**Description**

Default windows, strategy and smoothing functions used for portfolio backtesting.

**Usage**

```r
backtest_functions
```

### References

Würtz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
**Details**

**equidistWindows:**

Defines equal distant rolling windows.

The function requires two arguments: data and backtest, see above. To assign the horizon value to the backtest specification structure, use the function `setWindowSHorizon`.

**tangencyStrategy:**

A pre-defined tangency portfolio strategy.

The function requires four arguments: data, spec, constraints and backtest, see above.

**emaSmother:**

A pre-defined weights smoother (EMA) for portfolio backtesting.

The function requires three arguments: weights, spec and backtest, see above. To assign initial starting weights, smoothing parameter (lambda) or whether to perform double smoothing to the backtest specification structure, use the functions `setSmotherInitialWeights`, `setSmotherLambda` and `setSmotherDoubleSmoothing`, respectively.

**Value**

equidistWindows function returns the "from" and "to" dates of the rolling window in a list form.

tangencyStrategy function returns a S4 object of class "fPORTFOLIO".

emaSmother function returns a numeric vector of smoothed weights.

**References**

W"urtz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

**Description**

Extractor functions to get information from objects of class `fPFOLIOBACKTEST`.

**Arguments**

object an object of class `fPFOLIOBACKTEST` as returned by function `portfolioBacktest`. 

---

backtest-getMethods  Portfolio Backtest Extractors

---
References

W"urtz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Examples

```r
## portfolioBacktest Specification -
backtestSpec = portfolioBacktest()
backtestSpec

## Extract Windows Information -
getWindows(backtestSpec)
getWindowsFun(backtestSpec)
getWindowsParams(backtestSpec)
getWindowsHorizon(backtestSpec)

## Extract Strategy Information -
getStrategy(backtestSpec)
getStrategyFun(backtestSpec)
getStrategyParams(backtestSpec)

## Extract Smoother Information -
getSmoother(backtestSpec)
getSmootherFun(backtestSpec)
getSmootherParams(backtestSpec)
getSmootherLambda(backtestSpec)
getSmootherDoubleSmoothing(backtestSpec)
getSmootherInitialWeights(backtestSpec)
getSmootherSkip(backtestSpec)
```

---

**backtest-performance**  *Portfolio backtesting net performance*

**Description**

Displays plot of rebased portfolio performance and summary statistics.

**Usage**

```r
netPerformance(object, format = "%Y-%m-%d")
```

**Arguments**

- `object` a list, returned from running the function `portfolioSmoothing`.
- `format` a character string of the date format used
Value

A plot of rebased portfolio returns and tables summarising portfolio performance over time.

Note

This function will become obsolete by functions provided in the upcoming fPortfolioPerformance package.

References

W"urtz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Description

Creates and displays plots of cumulative assets returns, of portfolio weights, of rebalanced weights, of drawdowns and of a report summary for backtesting.

Usage

backtestPlot(object, which="all", labels=TRUE, legend=TRUE, at=NULL, format=NULL, cex=0.6, font=1, family="mono")

backtestAssetsPlot(object, labels=TRUE, legend=TRUE, at=NULL, format=NULL)
backtestWeightsPlot(object, labels=TRUE, legend=TRUE, at=NULL, format=NULL)
backtestRebalancePlot(object, labels=TRUE, legend=TRUE, at=NULL, format=NULL)
backtestPortfolioPlot(object, labels=TRUE, legend=TRUE, at=NULL, format=NULL)
backtestDrawdownPlot(object, labels=TRUE, legend=TRUE, at=NULL, format=NULL)
backtestReportPlot(object, cex=0.6, font=1, family="mono")

Arguments

- **object**: a list, returned from running the function portfolioSmoothing.
- **which**: an integer or string value. If the argument is an integer then it specifies which backtest plot should be displayed. If the argument take the character value all, which is the default, then all 6 available backtest plots will be displayed.
- **labels**: a logical flag, determining if the graph should be labeled automatically. This is the default case labels=TRUE. If set to FALSE then the graph will be displayed undecorated and the user can it decorate by himself.
legend a logical flag, determining if to the graph a legend should be added. This is
the default case labels=TRUE. If set to FALSE then the graph will be displayed
undecorated and the user can it decorate by himself.
at if NULL the time-axis ticks will be selected automatically. If at is a vector of
timeData character formatted dates then the axis ticks ar taken from this vector.
format if NULL the time-axis ticks are labeled automatically. If format is a POSIX
format string, then the label formats are taken from this string.
cex, font, family
font size, font and font family specification for the report.

Details

These backtest plot summarises the results obtained from portfolio backtesting.
The function backtestAssetsPlot displays the set of possible assets to construct a portfolio.
The function backtestWeightsPlot displays the recommended weights for investment.
The function backtestRebalancePlot displays the weight changes over time for individual assets
and for the portfolio.
The function backtestPortfolioPlot displays the daily, benchmark and portfolio series of a port-
folio backtest.
The function backtestDrawdownPlot displays the daily drawdowns for the benchmark and the
portfolio.
The function backtestReportPlot summarises the results from a portfolio backtest.

References

Würtz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmet-
rics eBook, Rmetrics Association and Finance Online, Zurich.

backtest-portfolio Portfolio backtesting

Description

Tests a portfolio by a rolling backtest.

Usage

portfolioBacktesting(formula, data, spec = portfolioSpec(),
constraints = "LongOnly", backtest = portfolioBacktest(),
trace = TRUE)

portfolioSmoothing(object, backtest, trace = TRUE)
Arguments

- **formula**: a formula describing the benchmark and assets used for backtesting in the form `backtest ~ assetA + ... + assetZ`. Here, `backtest` and `asset*` are column names of the data set.
- **data**: an object of class `timeseries`.
- **spec**: an S4 object of class `fpFOLIOSPEC` as returned by the function `portfoliospec`.
- **constraints**: a character string value or vector defining the constraints, for details we refer to `portfolioConstraints`.
- **backtest**: an S4 object of class `fpFOLIOBACKTEST` as returned by the function `portfoliobacktest`.
- **object**: a list as returned by the function `portfoliobacktesting`.
- **trace**: a logical flag, by default `TRUE`. Should the backtesting be traced?

References

Würtz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

```
backtest-specification

Specification of portfolio backtesting

Description

Specifies how the portfolio backtesting is performed.

Usage

```r
code
```

```
Arguments

windows a list, containing different arguments: windows, params (horizon).
strategy a list, containing different arguments: strategy, params.
smoother a list, containing different arguments: smoother, params. (doubleSmoothing, lambda, skip, initialWeights).
messages a list containing the backtesting messages.

Value

returns an S4 object of class "fPFOLIOBACKTEST".

References

W"urtz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

backtestStats 

Rolling portfolio backtesting statistics

Description

Computes rolling statistics for backtest analysis

Usage

backtestStats(object, FUN = "rollingSigma", ...)

rollingSigma(object)
rollingVaR(object)
rollingCVar(object)
rollingDaR(object)
rollingCDaR(object)

Arguments

object a list, returned from running the function portfolioSmoothing.
FUN a character string, specifying the name of the rolling statistics function.
... optional argument to be passed to the rolling statistics function FUN.

Details

The function rollingSigma calculates the portfolio risk, Sigma, over time.
The function rollingVaR calculates a rolling Value at Risk.
The function rollingCVar calculates a rolling Conditional Value at Risk.
The function rollingDaR calculates a rolling Drawdowns at Risk.
The function rollingCDaR calculates a rolling Conditional Drawdowns at Risk.
data-sets

References

W"urtz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

<table>
<thead>
<tr>
<th>data-sets</th>
<th>Assets Data Sets</th>
</tr>
</thead>
</table>

Description

Example data sets for portfolio optimization.

Usage

ECON85
ECON85LONG

GCCINDEX
SPISECTOR
SWX
LPP2005
SMALLCAP

GCCINDEX.RET
SPISECTOR.RET
SWX.RET
LPP2005.RET
SMALLCAP.RET

Value

an object of class "timeSeries".

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Description

Specifies portfolio backtesting objects.

Usage

```r
## S4 method for signature 'fpFOLIOBACKTEST'
show(object)
```

Arguments

- `object`: an S4 object of class `fpFOLIOBACKTEST`.

Details

**Portfolio Backtest Specification:**

The S4 class `fpFOLIOBACKTEST` specifies portfolio backtesting. The slots are:

- `@windows`: a list, setting the `windows` function that defines the rolling windows, and the set of window specific parameters `params`. E.g. The window horizon is set as a parameter `horizon = "24m`.

- `@strategy`: a list, setting the portfolio strategy to implement during the backtest, and any strategy specific parameters are found in `params`.

- `@smoother`: a list, specifying the smoothing style, given as a `smoother` function, and any smoother specific parameters are stored in the list `params`.

- `@messages`: a list, any messages collected during the backtest.

Value

`portfolioBacktest` returns an S4 object of class "fpFOLIOBACKTEST".

References

fPFOLIOCON

Portfolio Constraints Handling

Description

Creates a fPFOLIOCON object from string constraints.

Usage

```r
## S4 method for signature 'fPFOLIOCON'
show(object)
```

Arguments

- `object` - an object of class fPFOLIOCON as returned by the function `portfolioData`.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

fPFOLIODATA

Portfolio Data Handling

Description

Creates a fPFOLIODATA object with data set and statistical measures.

Usage

```r
portfolioData(data, spec = portfolioSpec())
```

Arguments

- `data` - [portfolioStatistics] - a time series or a named list, containing either a series of returns or named entries 'mu' and 'Sigma' being mean and covariance matrix.
- `object` - [show] - an object of class fPFOLIODATA as returned by the function `portfolioData`.
- `spec` - an S4 object of class fPFOLIOSPEC, the specification to be modified, by default the default of the function `portfolioSpec()`.
Details

**Dutch Portfolio Data Set:**

This data represents seven stocks from the Dutch AEX index, Netherlands blue chips. The data is a list of the covariance matrix and the return means and is based on daily returns over a period from January 1990 till end of October 2003. Companies representing the data are Elsevier, Fortis, Getronics, Heineken, Philips, Shell and Unilever.

**US Portfolio Data Set:**

The data inherits eight assets being indexes, commodities and bonds. The data is a time series of yearly returns from December 1973 till December 1994. Assets are TBills3m, LongBonds, SP500, Wilshire5000, NASDAQComp, LehmanBonds, EAFE, Gold.

**Simulated Mean-Cov Data Set:**

This data is taken from chapter 1.3.2 in Scherer, M., Martin, R.D. (2005); *Introduction To Modern Portfolio Optimization with NuOPT, S-PLUS and S+Bayes*, Springer, Berlin. It is a list of covariance matrix and the return means of imaginary assets. It is an example set for learning about optimization.

**World Index Returns Data Set:**

This data set is contributed by D. Locher (2007); It is a timeSeries object of four world index return data sets including Asia, Eastern Europe, Far East and Latin America.

**Value**

- `portfolioStatistics` returns a named list of estimated mean \( \mu \) and covariance \( \Sigma \) statistics, from a multivariate time series of assets.

- `portfolioData` returns a named list of the time series \( S \) and the portfolio \( S \) statistics as returned by the function `portfolioStatistics`.

**References**

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Description

Specifies portfolios.

Usage

```r
## S4 method for signature 'fPFOLIOSPEC'
show(object)
```

Arguments

- **object**: an S4 object of class fPFOLIOSPEC.

Details

Portfolio Specification Structure:

The S4 class fPFOLIOSPEC specifies the portfolio. The slots are:

- **@call**: a call, returning the matched function call.
- **@model**: a list, setting the type of portfolio to be optimized, and the mean/covariance estimator to be applied:
  - `type`: a character string denoting the type of portfolio, the implemented types are the Mean-Variance Markowitz Portfolio, "MV", and the Mean-CVaR Portfolio, "CVaR".
  - `estimator`: a vector of two character strings, the first denoting the mean estimator, and the second the covariance estimator. Additional meaningful selections include robust covariance estimators, e.g. c("mean","mcd"), or c("mean","shrink").
- **tailRisk**: a list() a list of optional tail risk information, currently not used.
- **params**: list() a list of optional model parameters, currently not used.
- **@portfolio**: a list, settings portfolio parameters including predefined weights, target return, risk free rate, number of frontier points:
  - `weights`: a numeric vector specifying the portfolio weights.
  - `targetReturn`: a numeric value specifying the target return. The default value sets the target return.
  - `targetRisk`: a numeric value specifying the target risk.
  - `targetAlpha`: a numeric value specifying the target alpha confidence level for CVaR portfolio optimization. The default value sets the target return.
  - `riskFreeRate`: a numeric value specifying the risk free rate.
  - `nFrontierPoints`: a numeric value determining the number of points on the efficient frontier.
- **@solver**: a list, setting the type of solver to be used for portfolio optimization:
  - `type`: a character string specifying the name of the solver to be used.
  - `trace`: a logical flag, should the optimization be traced?
- **@title**: a title string, with a default project title.
- **@description**: a character string, with a default project description.
Value

portfolioSpec

returns an S4 object of class "fPFOLOSPEC".

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

fPFOLOVAL

Values of Portfolio Frontiers

Description

Specifies portfolio Optimized Values.

Usage

```r
## S4 method for signature 'fPFOLOVAL'
show(object)
```

Arguments

object an S4 object of class fPFOLOVAL.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

fPORTFOLIO

Portfolio Class

Description

A collection and description of functions allowing to gain information about optimal portfolios. Generally, optimization is done via three arguments, data, specification of the portfolio, and constraints, while function portfolioFrontier has two additional arguments for title and description.
Usage

```r
## S3 method for class 'fPORTFOLIO'
plot(x, which = "ask", control = list(), ...)

## S3 method for class 'fPORTFOLIO'
summary(object, ...)
```

Arguments

- `control`: a list, defining the plotting parameters. The list modifies amongst others the color, e.g. `minvariance.col`, type of point, e.g. `tangency.pch`, or the dimension of the point, e.g. `cml.cex`, see Notes for a complete list of control parameters.
- `which`: which of the plots should be displayed? which can be either a character string, "all" (displays all plots) or "ask" (interactively asks which one to display), or a vector of integer values displaying the corresponding plot. Default value is "ask".
- `object, x`: an S4 object of class `fPORTFOLIO`.
- `...`: optional arguments to be passed.

Details

**Portfolio Class:**

This S4 class contains all information about the portfolio. Basically these are risk measure, mean and covariance estimation, target return, risk free rate, number of frontier points, ranges for calculation, see the "Value" section for a detailed description of the slots.

**Value**

`portfolioFrontier()` returns an S4 object of class "fPORTFOLIO", with the following slots:

- `@call`: a call, returning the matched function call.
- `@data`: a list with two named elements, `series` holding the time series data if available, otherwise NA, and `statistics`, itself a named list with two named elements `mu` and `Sigma` holding the vector of means and the matrix of covariances.
- `@description`: a character string, allowing for a brief project description.
- `@portfolio`: a list, containing parameter specifications for the portfolio:
  - `weights`: a numeric vector specifying the portfolio weights,
  - `targetReturn`: a numeric value specifying the target return,
  - `targetRisk`: a numeric value specifying the target risk,
  - `targetMean`: a numeric value specifying the target return determinated with function `mean()`,
  - `targetStdev`: a numeric value specifying the target risk in standard deviation as risk measure.
@specification a list with one named element spec which represents an object of class fPFOLIOSPEC, including all information about the portfolio specifications, see PortfolioSpec for further details.

@title a title string.

feasiblePortfolio
cmlPortfolio
tangencyPortfolio
minvariancePortfolio
efficientPortfolio
return an S4 object of class fPORTFOLIO having information only about one portfolio.

Control Parameters

In the following all elements of argument control from functions plot, weightsSlider, frontierSlider are listed.

sliderResolution [weightsSlider, frontierSlider] - a numeric, determining the numbers of slider points, by default nFrontierPoints/10.

sliderFlag [weightsSlider, frontierSlider] - a character string, denoting the slidertype, by default "frontier" for frontierSlider and "weights" for weightsSlider.

sharpeRatio.col [plot, frontierSlider] - a character string, defining color of the Sharpe ratio plot, by default "black".

minvariance.col a character string, defining color of the minimum variance portfolio, by default "red".

tangency.col a character string, defining color of the tangency portfolio, by default "steelblue".

cml.col [plot, frontierSlider] - a character string, defining color of the market portfolio and the capital market line, by default "green".

equalWeights.col [plot, frontierSlider] - a character string, defining the color of the equal weights portfolio, by default "blue".

runningPoint.col [weightsSlider] - a character string, defining color of the point indicating the current portfolio, by default "red".

singleAsset.col a character string vector, defining color of the single asset portfolios. The vector must have length the number of assets, by default rainbow.

twoAssets.col [plot, frontierSlider] - a character string, defining color of the two assets efficient frontier, by default "grey".

monteCarlo.col [plot, frontierSlider] - a character string, defining color of the Monte Carlo portfolios, by default "black".

minvariance.pch a number, defining symbol used for the minimum variance portfolio. See points for description. Default symbol is 17.

tangency.pch a number, defining symbol used for the tangency portfolio. See points for description. Default symbol is 17.

cml.pch [plot, frontierSlider] - a number, defining symbol used for the market portfolio. See points for description. Default symbol is 17.
equalWeights.pch [plot, frontierSlider] - a number, defining symbol used for the equal weights portfolio. See points for description. Default symbol is 15.

singleAsset.pch a number, defining symbol used for the single asset portfolios. See points for description. Default symbol is 18.

sharpeRatio.cex [plot, frontierSlider] - a number, determining size (percentage) of the Sharpe ratio plot, by default 0.1.

minvariance.cex a number, determining size (percentage) of the minimum variance portfolio symbol, by default 1.

tangency.cex a number, determining size (percentage) of the tangency portfolio symbol, by default 1.25.

cml.cex [plot, frontierSlider] - a number, determining size (percentage) of the market portfolio symbol, by default 1.25.

equalWeights.cex [plot, frontierSlider] - a number, determining size (percentage) of the equal weights portfolio symbol, by default 0.8.

runningPoint.cex [weightsSlider] - a number, determining size (percentage) of the point indicating the current portfolio equal weights portfolio symbol, by default 0.8.

singleAsset.cex a number, determining size (percentage) of the singel asset portfolio symbols, by default 0.8.

twoAssets.cex [plot, frontierSlider] - a number, determining size (percentage) of the two assets efficient frontier plot, by default 0.01.

monteCarlo.cex [plot, frontierSlider] - a number, determining size (percentage) of the Monte Carlo portfolio symbols, by default 0.01.

monteCarlo.cex [plot, frontierSlider] - a number, determining size (percentage) of the Monte Carlo portfolio symbols, by default 0.01.

mcSteps [plot] - a number, determining number of Monte Carlo portfolio, by default 5000.

pieR [plot, frontierSlider] - a vector, containing factors for shrinking and stretching the x- and y-axis, by default NULL, i.e. c(1, 1) is used. Default pie size is 1/15 of the plot range.

piePos [plot, frontierSlider] - a number, determining the weight on the efficient frontier, which is illustrated by the pie. Default is tangency portfolio

pieOffset [plot, frontierSlider] - a vector, containing the pie’s x- and y-axis offset from the efficient frontier. Default is NULL, i.e. the pie is set one default radius left of the efficient frontier.

xlim [weightsSlider, frontierSlider] - a vector, containing x-axis plot limits of the efficient frontier. Default setting is maximum of frontier range or single assets portfolios.

ylim [weightsSlider, frontierSlider] - a vector, containing y-axis plot limits of the efficient frontier. Default setting is maximum of frontier range or single assets portfolios.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Efficient Frontier Plot

Description

Plots the efficient frontier of an optimized portfolio and allows to add points and lines from specific portfolios.

Usage

frontierPlot(object, frontier = c("both", "lower", "upper"),
              col = c("black", "grey"), add = FALSE, labels = TRUE,
              return = c("mean", "mu"), risk = c("Cov", "Sigma", "CVaR", "VaR"),
              auto = TRUE, title = TRUE, ...)

minvariancePoints(object, return = c("mean", "mu"),
                   risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
cmlPoints(object, return = c("mean", "mu"),
           risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
cmlLines(object, return = c("mean", "mu"),
           risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
tangencyPoints(object, return = c("mean", "mu"),
                risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
tangencyLines(object, return = c("mean", "mu"),
               risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
equalWeightsPoints(object, return = c("mean", "mu"),
                    risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
singleAssetPoints(object, return = c("mean", "mu"),
                   risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
twoAssetsLines(object, return = c("mean", "mu"),
                risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)
sharpeRatioLines(object, return = c("mean", "mu"),
                  risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)

monteCarloPoints(object, mcSteps = 5000, return = c("mean", "mu"),
                   risk = c("Cov", "Sigma", "CVaR", "VaR"), auto = TRUE, ...)

tailoredFrontierPlot(object,
                      return = c("mean", "mu"), risk = c("Cov", "Sigma", "CVaR", "VaR"),
                      mText = NULL, col = NULL, xlim = NULL, ylim = NULL,
                      twoAssets = FALSE, sharpeRatio = TRUE, title = TRUE, ...)

Arguments

object an S4 object of class fPORTFOLIO, containing slots call, data, specification, constraints, portfolio, title, description.
frontier a character string, determining which part of the frontier should be extracted. "both" stands for the full hyperbola, "lower" for all points below the minimum variance return and "upper" for the actual efficient frontier, by default "both".

col a character string vector, setting the color. For frontierPlot it is a two dimensional a vector; first entry is the upper part of the frontier, second entry the lower, by default "black" and "grey". For the other functions the argument defines the color representation, by default sets the default color is the rainbow palette.

add a logical value, determining whether the frontier should be added to an existing plot, by default FALSE.

return a character string denoting which type of return should be plotted. Allowed values for the return are either "mean", or "mu".

risk a character string denoting which type of risk should be plotted. Allowed values for the risk measure are either "cov", "sigma", "VaR", or "CVaR".

auto a logical flag denoting if the type of return and risk to be plotted should be selected automatically, by default TRUE.

labels a logical flag, should the plot be automatically labeled and decorated? By default TRUE.

title a logical flag, should the plot obtain a default main title and x- and y-labels? By default TRUE.

mcSteps an integer value, the number of Monte Carlo steps.

xlim, ylim two numeric vectors with two elements, the plot range. If set to NULL the values for the plot ranges are determined automatically.

mText a character string, representing a marginal text string. If set to NULL the value is taken from the title of the input frontier argument.

twoAssets a logical flag, if TRUE, then the two assets frontier lines will be drawn.

sharpeRatio a logical flag, if TRUE, then the Sharpe ratio will be added to the plot.

... optional arguments to be passed.

Details

frontierPlot Plots efficient frontier,
minvariancePoints Adds minimum variance point,
cmplPoints Adds market portfolio,
cmlLines Adds capital market Line,
tangencyPoints Adds tangency portfolio point,
tangencyLines Adds tangency line,
equalWeightsPoints Adds point of equal weights portfolio,
singleAssetPoints Adds points of single asset portfolios,
twoAssetsLines Adds EF for all combinations of two assets,
sharpeRatioLines Adds Sharpe ratio line,
monteCarloPoints Adds randomly produced feasible portfolios,
tailoredFrontierPlot an example for a tailored plot.
References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

frontier-plotControl  Frontier Plot Control List

Description

Allows to modify plot settings for the frontier plot.

Usage

```r
frontierPlotControl(
  # Colors:
  sharpeRatio.col = "blue",
  minvariance.col  = "red",
  tangency.col     = "steelblue",
  cml.col          = "green",
  equalWeights.col = "blue",
  singleAsset.col  = "topo.colors",
  twoAssets.col    = "grey",
  monteCarlo.col   = "black",
  # Point Sizes:
  minvariance.cex  = 1.25,
  tangency.cex     = 1.25,
  cml.cex          = 1.25,
  equalWeights.cex = 1.25,
  singleAsset.cex  = 1.25,
  twoAssets.cex    = 0.01,
  monteCarlo.cex   = 0.01,
  sharpeRatio.cex  = 0.1,
  # Limits:
  xlim      = NULL,
  ylim      = NULL,
  # MC Steps:
  mcSteps   = 5000,
  # Pie Settings:
  pieR      = NULL,
  piePos    = NULL,
  pieOffset = NULL)
```
Arguments

sharpeRatio.col
  Color setting.
minvariance.col
  Color setting.
tangency.col
  Color setting.
cml.col
  Color setting.
equalWeights.col
  Color setting.
singleAsset.col
  Color setting.
twoAssets.col
  Color setting.
monteCarlo.col
  Color setting.
minvariance.cex
  Font point size setting.
tangency.cex
  Font point size setting.
cml.cex
  Font point size setting.
equalWeights.cex
  Font point size setting.
singleAsset.cex
  Font point size setting.
twoAssets.cex
  Font point size setting.
monteCarlo.cex
  Font point size setting.
sharpeRatio.cex
  Font point size setting.
xlim
  x-axis limit setting.
ylim
  y-axis limit setting.
mcSteps
  Number of Monte Carlo steps.
pieR
  Pie radius setting.
piePos
  Pie position coordinates setting.
pieOffset
  Pie offset coordinates setting.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
frontier-points  Get Frontier Points

Description

Extracts the risk and return coordinates of the efficient frontier.

Usage

frontierPoints(object, frontier = c("both", "lower", "upper"),
                 return = c("mean", "mu"), risk = c("Cov", "Sigma", "CVaR", "VaR"),
                 auto = TRUE)

Arguments

object  an object of class fPORTFOLIO.
frontier a character string denoting which part of the efficient portfolio should be extracted.
return character strings denoting which return measure should be plotted. Allowed values for the return are either "mean", or "mu".
risk character strings denoting which risk measure should be plotted. Allowed values for the risk measure are either "cov", "sigma", "VaR", or "CVaR".
auto a logical flag. If auto is TRUE, the default setting, then the risk will be identified automatically from the object.

Details

The automated risk detection, auto=TRUE takes the following decision:

    if (auto) {
        Type = getType(object)
        Estimator = getEstimator(object)
        if (Type == "MV") risk = "cov"
        if (Type == "MV" & Estimator != "covEstimator") risk = "sigma"
        if (Type == "QLPM") risk = "sigma"
        if (Type == "CVaR") risk = "CVaR"
    }

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Description

Mathematical Linear Programming.

Usage

rsolveLP(objective, lower=0, upper=1, linCons,
        control=list(solver="glpk", invoke=c("R", "AMPL", "NEOS")))

rglpkLP(objective, lower=0, upper=1, linCons, control=list())
        glpkLP
        glpkLPControl(solver = "glpk", project="r", trace=FALSE)

rsymphonyLP(objective, lower=0, upper=1, linCons, control=list())
        symphonyLP
        symphonyLPControl(solver="symphony", project="r", trace=FALSE)

ramplLP(objective, lower = 0, upper = 1, linCons, control=list())
        amplLP(objective, x_L=NULL, x_U=NULL, A=NULL, b_L=NULL, b_U=NULL,
        control=list())
        amplLPControl(solver="ipopt", project="ampl", inf=1e12, trace=FALSE)

rneosLP(objective, lower = 0, upper = 1, linCons, control=list())
        neosLP(objective, x_L=NULL, x_U=NULL, A=NULL, b_L=NULL, b_U=NULL,
        control=list())
        neosLPControl(solver="ipopt", category="lp", project="neos",
        inf=1e12, trace=FALSE)

Arguments

objective a numeric vector.
lower, upper lower and upper bounds.
linCons list of linear constraints: mat, lower, upper.
control control list.
x_L, x_U lower and upper box bounds.
A linear constraints matrix.
b_L, b_U lower and upper linear constraints bounds.
solver a character string, the solver name.
category a character string, the NEOS category name.
project a character string, the AMPL project name.
infix a numeric value, the maximum value used for bounds.
trace a logical flag, if TRUE the optimization will be traced.
Value

a list of class solver with the following named entries: opt, solution, objective, status, message, solver, version.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Mathematical Non-Linear Programming

Description

Mathematical Non-Linear Programming.

Usage

rdonlp2NLP(start, objective,
   lower=0, upper=1, linCons, funCons, control=list())

donlp2NLP(start, objective,
   par.lower=NULL, par.upper=NULL,
   eqA=NULL, eqA.bound=NULL,
   ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
   eqFun=list(), eqFun.bound=NULL,
   ineqFun=list(), ineqFun.lower=NULL, ineqFun.upper=NULL,
   control=list())

donlpNLPControl(
   iterma=4000, nstep=20, fnscale=1, report=FALSE, rep.freq=1,
   tau0=1, tau=0.1, del0=1, epsx=1e-05, delmin=0.1 * del0,
   epsdif=1e-08, nreset.multiplier=1, difftype=3, epsfcn=1e-16,
   taubnd=1, hessian=FALSE, te0=TRUE, te1=FALSE, te2=FALSE,
   te3=FALSE, silent=TRUE, intakt=TRUE)

rdonlp2

rsolnpNLP(start, objective,
   lower=0, upper=1, linCons, funCons, control=list())

solnpNLP(start, objective,
   par.lower=NULL, par.upper=NULL,
   eqA=NULL, eqA.bound=NULL,
   ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
   eqFun=list(), eqFun.bound=NULL,
   ineqFun=list(), ineqFun.lower=NULL, ineqFun.upper=NULL,
   control=list())

solnpNLPControl(
   rho=1, outer.iter=400, inner.iter=800, delta=1e-07, tol=1e-08, trace=0)

rsolnp
rnlinb2NLP(start, objective,
   lower=0, upper=1, linCons, funCons, control=list())
rnlinb2NLP(start, objective,
   par.lower=NULL, par.upper=NULL,
   eqA=NULL, eqA.bound=NULL,
   ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
   eqFun=list(), eqFun.bound=NULL,
   ineqFun=list(), ineqFun.lower=NULL, ineqFun.upper=NULL,
   control=list())
rnlinb2NLPControl(
   eval.max=500, iter.max=400, trace=0, abs.tol=1e-20, rel.tol=1e-10,
   x.tol=1.5e-08, step.min=2.2e-14, scale=1, R=1, beta.tol=1e-20)
rnlinb2

rampnlp(start, objective,
   lower=0, upper=1, amplCons, control=list(), ...)
ampLNLP()
ampLNLPControl(
   solver="minos", project="ampl", trace=FALSE)

Arguments

start a numeric vector, the start values.
objective a function object, the function to be optimized.
lower, upper lower and upper bounds.
linCons list of linear constraints: mat, lower, upper.
funCons list of function constraints.
amplCons AMPL constraints.
control control list.
... optional arguments to be passed.
par.lower, par.upper ...
eqA ...
eqA.bound ...
ineqA ...
ineqA.lower, ineqA.upper ...
eqFun ...
eqFun.bound ...
ineqFun ...
ineqFun.lower, ineqFun.upper ...
iterma 4000
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nstep</td>
<td>20</td>
</tr>
<tr>
<td>fnscale</td>
<td>1</td>
</tr>
<tr>
<td>report</td>
<td>FALSE</td>
</tr>
<tr>
<td>rep.freq</td>
<td>1</td>
</tr>
<tr>
<td>tau0</td>
<td>1</td>
</tr>
<tr>
<td>tau</td>
<td>0.1</td>
</tr>
<tr>
<td>del0</td>
<td>1</td>
</tr>
<tr>
<td>epsx</td>
<td>1e-5</td>
</tr>
<tr>
<td>delmin</td>
<td>0.1 * del0</td>
</tr>
<tr>
<td>epsdif</td>
<td>1e-8</td>
</tr>
<tr>
<td>nreset.multiplier</td>
<td>1</td>
</tr>
<tr>
<td>difftype</td>
<td>3</td>
</tr>
<tr>
<td>epsfcn</td>
<td>1e-16</td>
</tr>
<tr>
<td>taubnd</td>
<td>1</td>
</tr>
<tr>
<td>hessian</td>
<td>FALSE</td>
</tr>
<tr>
<td>te0</td>
<td>TRUE</td>
</tr>
<tr>
<td>te1</td>
<td>FALSE</td>
</tr>
<tr>
<td>te2</td>
<td>FALSE</td>
</tr>
<tr>
<td>te3</td>
<td>FALSE</td>
</tr>
<tr>
<td>silent</td>
<td>TRUE</td>
</tr>
<tr>
<td>intakt</td>
<td>TRUE</td>
</tr>
<tr>
<td>rho</td>
<td>1</td>
</tr>
<tr>
<td>outer.iter</td>
<td>400</td>
</tr>
<tr>
<td>inner.iter</td>
<td>800</td>
</tr>
<tr>
<td>delta</td>
<td>1.0e-7</td>
</tr>
<tr>
<td>tol</td>
<td>1.0e-8</td>
</tr>
<tr>
<td>eval.max</td>
<td>500</td>
</tr>
<tr>
<td>iter.max</td>
<td>400</td>
</tr>
<tr>
<td>trace</td>
<td>0</td>
</tr>
<tr>
<td>abs.tol</td>
<td>1e-20</td>
</tr>
<tr>
<td>rel.tol</td>
<td>1e-10</td>
</tr>
<tr>
<td>x.tol</td>
<td>1.5e-08</td>
</tr>
<tr>
<td>step.min</td>
<td>2.2e-14</td>
</tr>
<tr>
<td>scale</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
</tr>
<tr>
<td>beta.tol</td>
<td>1e-20</td>
</tr>
<tr>
<td>solver</td>
<td>solver name</td>
</tr>
<tr>
<td>project</td>
<td>project name</td>
</tr>
</tbody>
</table>
Value

a list of class solver with the following named entries: opt, solution, objective, status, message, solver, version.

References


Description

Mathematical Quadratic Programming.

Usage

rsolveQP(objective, lower=0, upper=1, linCons,
        control=list(solver="quadprog", invoke=c("R", "AMPL", "NEOS")))

rquadprogQP(objective, lower=0, upper=1, linCons, control=list())
quadprogQP(objective=list(dvec=NULL, Dmat=NULL),
          par.lower=NULL, par.upper=NULL,
          eqA=NULL, eqA.bound=NULL,
          ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
          control=list())
quadprogQPControl(solver="quadprog", trace=FALSE)
quadprog

ripopQP(objective, lower=0, upper=1, linCons, control=list())
ipopQP(objective=list(dvec=NULL, Dmat = NULL),
        par.lower=NULL, par.upper=NULL,
        eqA=NULL, eqA.bound=NULL,
        ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
        control=list())
ipopQPControl(
    sigf=12, maxiter=400, margin=0.05, bound=10, verb=0,
    inf=1e12, solver="ipop", trace=FALSE)
ripop

ramplQP(objective, lower=0, upper=1, linCons, control=list())
amplQP(objective=list(dvec=NULL, Dmat=NULL),
        x_L=NULL, x_U=NULL, A=NULL, b_L=NULL, b_U=NULL,
        control=list(), ...)
amplQPControl(solver="ipopt", project="ampl",
              inf=1e12, trace = FALSE)
rkestrelQP(objective, lower=0, upper=1, linCons, control=list())
kestrelQP(objective=list(dvec=NULL, Dmat=NULL),
    x_L=NULL, x_U=NULL, A=NULL, b_L=NULL, b_U=NULL,
    control=list(), ...)
kestrelQPControl(solver="loqo", project="kestrel",
    inf=1e12, trace = FALSE)

rneosQP(objective, lower=0, upper=1, linCons, control=list())
neosQP(objective=list(dvec=NULL, Dmat=NULL),
    x_L=NULL, x_U=NULL, A=NULL, b_L=NULL, b_U=NULL,
    control=list(), ...)
eosQPControl(solver="ipopt", category="nco", project="neos",
    inf=1e12, trace=FALSE)

Arguments

  objective       ...  
  lower, upper    lower and upper bounds.  
  linCons         list of linear constraints: mat, lower, upper.  
  control         control list.  
                   ... optional arguments to be passed.  
  par.lower, par.upper
                   ...  
  eqA             ...  
  eqA.bound       ...  
  ineqA           ...  
  ineqA.lower, ineqA.upper
                   ...  
  x_L, x_U        ...  
  A               ...  
  b_L, b_U        ...  
  solver          ...  
  category        ...  
  project         ...  
  inf             ...  
  trace           ...  
  sigf            ...  
  maxiter         ...  
  margin          ...  
  bound           ...  
  verb            ...
Value

a list of class solver with the following named entries: opt, solution, objective, status, message, solver, version.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

**methods-plot**

*plot-methods*

**Description**

plot-methods.

**References**

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

**methods-show**

*Portfolio Print Methods*

**Description**

show-methods.

**Usage**

```r
# S4 method for signature 'fPORTFOLIO'
show(object)
```

**Arguments**

- `object`: an S4 object of class `fPORTFOLIO`.

**References**

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Description

Functions for time series aggregation, converting a time series from a daily to a monthly or weekly base.

Usage

stabilityAnalytics(index, method=c("turns", "drawdowns", "garch", "riskmetrics", "bcp", "pcout"), ...)

turnsAnalytics(index, spar=0.5, main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y")
drawdownsAnalytics(index, spar=0.5, main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y")
garchAnalytics(index, spar = 0.5, main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y")
riskmetricsAnalytics(index, spar=0.5, lambda=0.9, main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y")
bcpAnalytics(index, spar=0.5, FUN=returns, method=c("prob", "mean", "var"), main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y")
pcoutAnalytics(index, spar=0.5, main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y", strong=TRUE, k=2, cs=0.25, outbound=0.25)

addRainbow(analytics, palette=rainbow, a=0.3, b=0.8, K=100)

waveletSpectrum(index, spar=0.5, main=NULL, trace=TRUE, doplot=TRUE, at=pretty(index), format="%m/%y")

parAnalytics()
Portfolio Constraints

Description

Computes portfolio constraints given constraints strings.

Usage

portfolioConstraints(data, spec=portfolioSpec(), constraints="LongOnly", ...)

minWConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
maxWConstraints(data, spec=portfolioSpec(), constraints="LongOnly")

eqsumWConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
minsumWConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
maxsumWConstraints(data, spec=portfolioSpec(), constraints="LongOnly")

minBConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
maxBConstraints(data, spec=portfolioSpec(), constraints="LongOnly")

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Arguments

constraints: a character value or character vector, containing the constraint strings. Setting constraints is described in the details section.

data: a list, having a statistics named list, having named entries 'mu' and 'Sigma', containing the information of the statistics.

spec: an S4 object of class `fpFOL10SPEC` as returned by the function `portfolioSpec`.

... arguments passed to the function `.setRdonly2Constraints`. For internal use only.

Details

**How to define constraints?**

Constraints are defined by a character string or a vector of character strings.

*Summary Constraints:* NULL, "LongOnly", "Short"

There are three special cases, the settings `constraints=NULL`, `constraints="Short"`, and `constraints="LongOnly"`. Note, that these three constraint settings are not allowed to be combined with more general constraint definitions.

NULL: This selection defines the default value and is equivalent to the "LongOnly" case, see below.

"Short": This selection defines the case of unlimited short selling, i.e. each weight may range between -Inf and Inf. Consequently, there are no group constraints. Risk budget constraints are not included in the portfolio optimization.

"LongOnly": This selection is the same as the default setting. Each weight may range between 0 ans 1. No group constraints and risk budget constraints will be included in the portfolio optimization.

*Lower and Upper Bounds: minW and maxW*

*Group Constraints: eqsumW, minsumW and maxsumW*

Lower and upper bounded portfolios may be specified by a vector of character strings which describe executable code, setting values to to vectors `minW`, `maxW`, `minsumW`, and `maxsumW`. The individual string elements of the vector have the following form:

*box constraints* "minW[Asset(s)]=Value(s)", and/or

"maxW[Asset(s)]=Value(s)".
sector constraints "\( \text{sumw}[\text{Asset(s)}] = \text{Value(s)} \)”, and/or
"\( \text{maxsumw}[\text{Asset(s)}] = \text{Value(s)} \)”. 

Asset(s) is an index of one or more assets, and value a numeric value or vector assigning the desired value. Note, if the values range between zero and one, then we have a long only portfolio allowing for box and group constraints of the weights. If the values are set to negative values, and values larger than one, then (constrained) short selling will be allowed.

Risk Budget Constrained Portfolios:

By default, risk budgets are not included in the portfolio optimization. Covariance risk budgets have to be added explicitly, and have the following form:

box constraints "\( \text{minB}[\text{Asset(s)}] = \text{Value(s)} \)”, and/or
"\( \text{minB}[\text{Asset(s)}] = \text{Value(s)} \)”. 

Again, Asset(s) is an index of one or more assets, and value a numeric value or vector with numbers ranging between zero and one, assigning the desired risk budgets.

Note, risk budget constraints will enforce diversification at the expense of return generation. The resulting portfolios will thus lie below the unconstrained efficient frontier.

Non-Linear Constraints: \( \text{listF}, \text{minF}, \text{maxF} \)

Value

an object of class S4.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

portfolio-covEstimator

*Covariance Estimators*

Description

Functions to estimate and robustify the sample mean and covariance of rectangular objects.

Usage

covEstimator(x, spec = NULL, ...)
mveEstimator(x, spec = NULL, ...)
mcdEstimator(x, spec = NULL, ...)

lpmEstimator(x, spec = NULL, ...)
slpmEstimator(x, spec = NULL, ...)
kendallEstimator(x, spec = NULL, ...)
spearmanEstimator(x, spec = NULL, ...)

covMcdEstimator(x, spec = NULL, ...)

covOGKEstimator(x, spec = NULL, ...)

shrinkEstimator(x, spec = NULL, ...)

nnveEstimator(x, spec = NULL, ...)

**Arguments**

- **x**: an object of class `timeSeries`.
- **spec**: unused, may be used to pass information from the portfolio specification object to the mean and covariance estimator function.
- **...**: optional arguments to be passed to the underlying estimators.

**Details**

The functions are underlying the following algorithms:

- `covEstimator` uses standard covariance estimation,
- `mveEstimator` uses the function "cov.mve" from the MASS package,
- `mcdEstimator` uses the function "cov.mcd" from the MASS package,
- `lpmEstimator` returns lower partial moment estimator,
- `kendallEstimator` returns Kendall’s rank estimator,
- `spearmanEstimator` returns Spearman’s rank estimator,
- `covMcdEstimator` requires "covMcd" from package robustbase,
- `covOGKEstimator` requires "covOGK" from package robustbase,
- `nnveEstimator` uses built-in from package covRobust,
- `shrinkEstimator` uses built-in from package corpcor.

**Value**

the functions return a list with two entries named `mu` and `Sigma`. The first denotes the vector of column means, and the second the covariance matrix. Note, that the output of this function can be used as data input for the portfolio functions to compute the efficient frontier.

**Author(s)**

- ... for R’s MASS package,
- ... for R’s robustbase package,
- ... for R’s covRobust package,
- Juliane Schaefer and Korbinian Strimmer for R’s corpcor package,
- Diethelm Wuertz for this Rmetrics port.

**References**


Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

**portfolio-dataSets**  
**portfolioData2**

**Description**  
portfolioData2.

**References**  
Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

**portfolio-efficientPortfolio**  
*Efficient Portfolios*

**Description**  
Returns efficient portfolios.

**Usage**  

```r
efficientPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
maxratioPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
tangencyPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
minriskPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
minvariancePortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
maxreturnPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
```

**Arguments**

- **constraints**  
a character string vector, containing the constraints of the form  
"minW[asset]=percentage" for box constraints resp.  
"maxsumW[assets]=percentage" for sector constraints.

- **data**  
a multivariate time series described by an S4 object of class `timeSeries`. If your `timeSeries` is not a `timeSeries` object, consult the generic function `as.timeSeries` to convert your time series.

- **spec**  
an S4 object of class `fPFOLIOSPEC` as returned by the function `portfolioSpec`.
Details

Efficient Portfolio:

An efficient portfolio is a portfolio which lies on the efficient frontier. The efficientPortfolio function returns the properties of the efficient portfolio as an S4 object of class fPORTFOLIO.

Minimum Risk or Tangency Portfolio:

The function tangencyPortfolio returns the portfolio with the highest return/risk ratio on the efficient frontier. For the Markowitz portfolio this is the same as the Sharpe ratio. To find this point on the frontier the return/risk ratio calculated from the target return and target risk returned by the function efficientPortfolio.

Global minimum risk or Minimum Variance Portfolio:

The function minvariancePortfolio returns the portfolio with the minimal risk on the efficient frontier. To find the minimal risk point the target risk returned by the function efficientPortfolio is minimized.

Maximum Return Portfolio:

The function maxreturnPortfolio returns the portfolio with the maximal return for a fixed target risk.

Value

returns an S4 object of class "fPORTFOLIO".

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Usage

feasiblePortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
Arguments

- **constraints**: a character string vector, containing the constraints of the form "minW[asset]=percentage" for box constraints resp. "maxW[assets]=percentage" for sector constraints.
- **data**: a multivariate time series described by an S4 object of class `timeSeries`. If your `timeSerie` is not a `timeSeries` object, consult the generic function `as.timeSeries` to convert your time series.
- **spec**: an S4 object of class `fPFOLIOSPEC` as returned by the function `portfolioSpec`.

Details

A feasible portfolio is a portfolio with given weights which lies inside the feasible region of portfolios.

The function requires three arguments: `data`, `spec` (specifications), and `constraints`, see above. Be sure that the specification structure "spec" has defined a weights vector which is different from "NULL". To assign values to the weights in the specification structure, use the function `setWeights`.

The `feasiblePortfolio` function returns the properties of the feasible portfolio as an S4 object of class `fPORTFOLIO`.

Value

The `feasiblePortfolio` function returns an S4 object of class "fPORTFOLIO".

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

**portfolio-getData**  
*Portfolio Data Extractor Functions*

**Description**

Extracts information from an object of class `fPFOLIODATA`.

**Usage**

```r
## S3 method for class 'fPFOLIODATA'
getData(object)
## S3 method for class 'fPFOLIODATA'
getSeries(object)
## S3 method for class 'fPFOLIODATA'
getNAssets(object)
## S3 method for class 'fPFOLIODATA'
getUnits(x)
```
## S3 method for class 'FPFOLIODATA'
getStatistics(object)
## S3 method for class 'FPFOLIODATA'
getMean(object)
## S3 method for class 'FPFOLIODATA'
getCov(object)
## S3 method for class 'FPFOLIODATA'
getMu(object)
## S3 method for class 'FPFOLIODATA'
getSigma(object)
## S3 method for class 'FPFOLIODATA'
getEstimator(object)
## S3 method for class 'FPFOLIODATA'
getTailRisk(object)

### Arguments

- **object**: an object of class `FPFOLIODATA`.
- **x**: an object of class `FPFOLIODATA`.

### Details

- `getData`: Extracts data slot,
- `getSeries`: Extracts assets series,
- `getNAssets`: Extracts number of assets,
- `getUnits`: Extracts names of assets,
- `getStatistics`: Extracts statistics slot,
- `getMean`: Extracts mean vector,
- `getCov`: Extracts covariance matrix,
- `getMu`: Extracts mu vector,
- `getSigma`: Extracts Sigma matrix,
- `getEstimator`: Extracts Sigma matrix,
- `getTailRisk`: Extracts tail risk slot.

### References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Description

Extractor functions to get information from objects of class fPFOLIODATA, fPFOLIOSPEC, fPFOLIODATA, fPFOLIOVAL, and fPORTFOLIO.

Usage

getConstraints(object)
getControl(object)
getCov(object)
getCovRiskBudgets(object)
gedata(object)
getEstimator(object)
getMean(object)
getMu(object)
getNAssets(object)
getNFrontierPoints(object)
getObjective(object)
getOptim(object)
getOptions(object)
getOptimize(object)
getPortfolio(object)
getParams(object)
getRiskFreeRate(object)
getSeries(object)
getSigma(object)
getSolver(object)
getSpec(object)
getStatistics(object)
getStatus(object)
getAlpha(object)
getTailRisk(object)
getTailRiskBudgets(object)
getTargetReturn(object)
getTargetRisk(object)
getTrace(object)
getType(object)
getWeights(object)

Arguments

object an object of class fPFOLIODATA, fPFOLIOSPEC or fPORTFOLIO.

... optional arguments to be passed.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Portfolio Class Extractors

Description

A collection and description of functions allowing to get information about an object of class fPORTFOLIO.

The functions are:

- `getData` Extracts ...
- `getSeries` Extracts ...
- `getStatistics` Extracts ...
- `getNAssets` Extracts ...
- `getSpec` Extracts ...
- `getType` Extracts ...
- `getEstimator` Extracts ...
- `getParams` Extracts ...
- `getSolver` Extracts ...
- `getTrace` Extracts ...
- `getConstraints` Extracts ...
- `getPortfolio` Extracts ...
- `getWeights` Extracts ...
- `getTargetReturn` Extracts ...
- `getTargetRisk` Extracts ...
- `getAlpha` Extracts ...
- `getRiskFreeRate` Extracts ...
- `getNFrontierPoints` Extracts ...
- `getStatus` Extracts ...
- `getCovRiskBudgets` Extracts ...
- `getTailRiskBudgets` Extracts ...

Usage

```r
## S3 method for class 'fPORTFOLIO'
getData(object)
## S3 method for class 'fPORTFOLIO'
getSeries(object)
## S3 method for class 'fPORTFOLIO'
getNAssets(object)
## S3 method for class 'fPORTFOLIO'
getUnits(x)
## S3 method for class 'fPORTFOLIO'
getStatistics(object)
## S3 method for class 'fPORTFOLIO'
```
getMean(object)
    ## S3 method for class 'fPORTFOLIO'
getCov(object)
    ## S3 method for class 'fPORTFOLIO'
getMu(object)
    ## S3 method for class 'fPORTFOLIO'
getSigma(object)
    ## S3 method for class 'fPORTFOLIO'
getEstimator(object)
    ## S3 method for class 'fPORTFOLIO'
getSpec(object)
    ## S3 method for class 'fPORTFOLIO'
getModel(object)
    ## S3 method for class 'fPORTFOLIO'
getType(object)
    ## S3 method for class 'fPORTFOLIO'
getOptimize(object)
    ## S3 method for class 'fPORTFOLIO'
getEstimator(object)
    ## S3 method for class 'fPORTFOLIO'
getTailRisk(object)
    ## S3 method for class 'fPORTFOLIO'
getParams(object)
    ## S3 method for class 'fPORTFOLIO'
getOptim(object)
    ## S3 method for class 'fPORTFOLIO'
getSolver(object)
    ## S3 method for class 'fPORTFOLIO'
getTrace(object)
    ## S3 method for class 'fPORTFOLIO'
getConstraints(object)
    ## S3 method for class 'fPORTFOLIO'
getPortfolio(object)
    ## S3 method for class 'fPORTFOLIO'
getWeights(object)
    ## S3 method for class 'fPORTFOLIO'
getTargetReturn(object)
    ## S3 method for class 'fPORTFOLIO'
getTargetRisk(object)
    ## S3 method for class 'fPORTFOLIO'
getAlpha(object)
    ## S3 method for class 'fPORTFOLIO'
getRiskFreeRate(object)
    ## S3 method for class 'fPORTFOLIO'
getNFrontierPoints(object)
## S3 method for class 'fPORTFOLIO'
getStatus(object)

## S3 method for class 'fPORTFOLIO'
getCovRiskBudgets(object)
## S3 method for class 'fPORTFOLIO'
getTailRiskBudgets(object)

## S3 method for class 'fPORTFOLIO'
getA(object)
## S3 method for class 'fPORTFOLIO'
getControl(object)
## S3 method for class 'fPORTFOLIO'
getObjective(object)
## S3 method for class 'fPORTFOLIO'
getOptions(object)

### Arguments

- **object**
  an object of class fPORTFOLIO, containing slots call, data, specification, constraints, portfolio, title, description.

- **x**
  an object of class fPORTFOLIO, containing slots call, data, specification, constraints, portfolio, title, description.

### References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics e-Book, Rmetrics Association and Finance Online, Zurich.
getEstimator(object)
## S3 method for class 'FPFOLIOSPEC'
getTailRisk(object)
## S3 method for class 'FPFOLIOSPEC'
getParams(object)
## S3 method for class 'FPFOLIOSPEC'
getPortfolio(object)
## S3 method for class 'FPFOLIOSPEC'
getWeights(object)
## S3 method for class 'FPFOLIOSPEC'
getTargetReturn(object)
## S3 method for class 'FPFOLIOSPEC'
getTargetRisk(object)
## S3 method for class 'FPFOLIOSPEC'
getAlpha(object)
## S3 method for class 'FPFOLIOSPEC'
getRiskFreeRate(object)
## S3 method for class 'FPFOLIOSPEC'
getNFrontierPoints(object)
## S3 method for class 'FPFOLIOSPEC'
getStatus(object)
## S3 method for class 'FPFOLIOSPEC'
getOptim(object)
## S3 method for class 'FPFOLIOSPEC'
getSolver(object)
## S3 method for class 'FPFOLIOSPEC'
getObjective(object)
## S3 method for class 'FPFOLIOSPEC'
getOptions(object)
## S3 method for class 'FPFOLIOSPEC'
getControl(object)
## S3 method for class 'FPFOLIOSPEC'
getTrace(object)
## S3 method for class 'FPFOLIOSPEC'
getMessages(object)

Arguments

object an object of class fPFOLIOSPEC.

Details

ggetType Extracts portfolio type from specification,
getOptimize Extracts what to optimize from specification,
getEstimator Extracts type of covariance estimator,
getTailRisk Extracts list of tail dependency risk matrixes,
getParams  Extracts parameters from specification,
getWeights  Extracts weights from a portfolio object,
getTargetReturn  Extracts target return from specification,
getTargetRisk  Extracts target risks from specification,
getAlpha  Extracts target VaR-alpha specification,
getRiskFreeRate  Extracts risk free rate from specification,
getNFrontierPoints  Extracts number of frontier points,
getStatus  Extracts the status of optimization,
getSolver  Extracts solver from specification,
getObjective  Extracts name of objective function,
getTrace  Extracts solver’s trace flag.

References
Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Arguments

- **object**: an object of class `fPFOLIODATA`.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Description

Computes covariance and CVaR portfolio risk.

Usage

```r
covRisk(data, weights)
varRisk(data, weights, alpha = 0.05)
cvarRisk(data, weights, alpha = 0.05)
```

Arguments

- **data**: a multivariate time series described by an S4 object of class `timeSeries`.
- **weights**: a numeric vector of weights.
- **alpha**: a numeric value, the confidence level, by default `alpha=0.05`, i.e. 5%.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Description

Computes the efficient portfolio frontier.

Usage

```r
portfolioFrontier(data, spec = portfolioSpec(), constraints = "LongOnly",
                   include.mvl = TRUE, title = NULL, description = NULL)
```
Arguments

- **constraints**: a character string vector, containing the constraints of the form "minW[asset]=percentage" for box constraints resp. "maxsumW[assets]=percentage" for sector constraints.
- **data**: a multivariate time series described by an S4 object of class `timeseries`. If your `timeSerie` is not a `timeseries` object, consult the generic function `as.timeSeries` to convert your time series.
- **description**: a character string which allows for a brief description.
- **include.mvl**: a logical flag, should the minimum variance locus be added to the plot?
- **spec**: an S4 object of class `fpFOLi0SPEC` as returned by the function `portfolioSpec`.
- **title**: a character string which allows for a project title.

Details

**Portfolio Frontier:**

The function `portfolioFrontier` calculates the whole efficient frontier. The portfolio information consists of five arguments: data, specifications, constraints, title and description.

The range of the frontier is determined from the range of the asset returns, and the number of equidistant points in the returns, is calculated from the number of frontier points hold in the specification structure. To extract or to modify the number of frontier points use the functions `getNFrontierPoints` and `setNFrontierPoints`.

The `frontierPortfolio` function returns the properties of the the efficient frontier as an S4 object of class `fPORTFOLIO`.

Value

The `portfolioFrontier` function returns an S4 object of class "fPORTFOLIO".

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

**portfolio-portfolioSpec**

*Specification of Portfolios*

**Description**

Specifies a portfolio from scratch.
Usage

```r
portfolioSpec(
  model = list(
    type = "MV", optimize = "minRisk",
    estimator = "covEstimator", tailRisk = list(),
    params = list(alpha = 0.05, a = 1)),
  portfolio = list(
    weights = NULL, targetReturn = NULL,
    targetRisk = NULL, riskFreeRate = 0, nFrontierPoints = 50,
    status = NA),
  optim = list(
    solver = "solveRquadprog",
    objective = c("portfolioObjective", "portfolioReturn", "portfolioRisk"),
    options = list(meq = 2), control = list(), trace = FALSE),
  messages = list(
    messages = FALSE, note = ""),
  ampl = list(
    ampl = FALSE, project = "ampl", solver = "ipopt",
    protocol = FALSE, trace = FALSE)
)
```

Arguments

- **model**: a list, containing different arguments: type, estimator, params. See these arguments for further explanation.
- **portfolio**: a list, containing different arguments: weights, targetReturn, riskFreeRate, nFrontierPoints. See these arguments for further explanation.
- **optim**: a list with four entries, a character string solver denoting the type of the solver to be used, a params list to pass further arguments to the objective function to optimize, a control list for all control settings of the solver, and a logical flag, trace denoting if the optimization should be traced.
- **messages**: a list, for optional messages.
- **ampl**: a list, controls settings for the R/AMPL interface.

Details

To optimize a portfolio of assets we first have to specify it. All settings which specify a portfolio of assets are represented by a S4 class named `fpFOLIOSPEC`.

```r
setClass("fpFOLIOSPEC",
  representation(
    model = "list",
    portfolio = "list",
    optim = "list")
)```
An object of class `fpFOLIOSPEC` has three slots, named `@model`, `@portfolio`, and `@optim`. The first slot `@model` holds the model information, the second slot `@portfolio` the portfolio information, and the last slot `@optim` the information about the solver used for optimization.

The default settings are as follows:

```r
model = list(
  type = "MV",
  optimize = "minRisk",
  estimator = "covEstimator",
  tailRisk = list(),
  params = list(alpha = 0.05, a = 2)),
portfolio = list(
  weights = NULL,
  targetReturn = NULL,
  targetRisk = NULL,
  riskFreeRate = 0,
  nFrontierPoints = 50,
  status = NA),
optim = list(
  solver = "solveRquadprog",
  objective = NULL,
  parames = list(),
  control = list(meq = 2),
  trace = FALSE)
```

**Model Slot:**

*Type of Model:*

The list entry `type` from the `@model` slot describes the type of the desired portfolio. The current implementation supports three types of portfolios. This may be a Markowitz mean – variance portfolio named "MV", a mean – lower partial moment portfolio named "LPM", or a mean – CVaR conditional value-at-risk portfolio named "CVaR". One can use the function `getType` to retrieve the current setting and the function `setModelType` to modify this selection.

*What to optimize?*

The list entry `optimize` from the `@model` slot describes what should be optimized. Two choices are possible: Either

```r
\code{"minRisk"}
```

which minimizes the risk if the target returns is given, or

```r
\code{"maxReturn"}
```

which maximizes the return if the target risk is given. One can use the function `getOptimize` to retrieve the current setting and the function `setOptimize` to modify this selection.
How to estimate mean and covariance?

The list entry estimator from the `@model` slot requests for a string that denotes the function name of the covariance estimator which should be used for the estimation of risk.

In Markowitz’ mean-variance portfolio model, type="MV", the default function

```
\code("covEstimator")
```

is used which computes the standard column means of the multivariate assets data series and the standard covariance matrix. Alternative robust estimators include

```
\code("covMcdEstimator")
\code("covOGKEstimator")
\code("mveEstimator")
\code("nnveEstimator")
\code("mcdEstimator")
```

In addition a shrinkage covariance estimator named

```
\code("shrinkEstimator")
```

and a bagged covariance estimator named

```
\code("baggedEstimator")
```

are also available. Note, the experienced user can add his own function to estimate in any alternative way the mean and the covariance of the multivariate assets data series. In this case (s)he has to write a function, e.g. named

```
\code{myEstimator=function(x,spec=NULL,...})
```

where `x` is a multivariate time series, spec optionally the portfolio specification, if required, and ... additional arguments passed to the users code. Note, `myEstimator` must a return a named list, with at least the following two entries `$\mu$` and `$\Sigma$`, which represent estimators for the mean and covariance, respectively.

In the case of the Mean – Lower-Partial-Moment portfolio, type="LPM" we make use of the equivalence to Markowitz’ mean-variance portfolio with a modified covariance estimator, i.e.

```
\code("lpmEstimator")
```
Note, in this case the setting of type="LPM" changes the covariance estimator function name from any selection previously made to the function automatically to "lpmEstimator" which returns the LPM mean and covariance estimates.

One can use the function getEstimator to retrieve the current setting and the function setEstimator to modify this selection.

Tail Risk List:
The list entry tailRisk from the @model slot is an empty list. It can be used to add tail risk budget constrains to the optimization. In this case a square matrix of the size of the number of assets is expected as list entry, which contains bivariate tail risk measures, i.e. the tail dependence coefficients estimated via a copulae approach. Use the function setEstimator to modify this selection.

The list entry parameters from the @model slot is a list with additional parameters used in different situations. It can be enhanced by the user if needed. By default it contains the exponent a=2, the parameter needed for "LPM" portfolio optimization, and it contains the targetAlpha=0.05, the confidence level for "CVaR" portfolio optimization. Use the function setParams to modify this selection.

Portfolio Slot:
The values weights, targetReturn, and targetRisk from the portfolio slot have to be considered in common. By default all three are set to NULL. If this is the case, then it is assumed that an equal weight portfolio should be calculated. If only one of the three values is different from NULL then the following procedure will be started. If the weights are specified then it is assumed that a feasible portfolio should be considered. If the target return is fixed then it is assumed that the efficient portfolio with the minimal risk will be considered. And finally if the risk is fixed, then the return should be maximized. Use the functions setWeights, setTargetReturn, and setTargetRisk to modify this selection. Note, the change in of the three functions will influence the settings of the other two.

The riskFreeRate=0 is also stored in the portfolio slot. Its value defaults to zero. It can be changed by the user. Use the function setRiskFreeRate to modify this selection.

The number of frontier points required by the calculation of the portfolioFrontier is obtained from the value of nFrontierPoints=50 hold in the portfolio slot. Its value defaults to 50. It can be changed by the user. Use the function setNFrontierPoints to modify this selection.

The final status of portfolio optimization is returned and stored in the portfolio slot. Before optimization the value is unset to NA, after optimization a value of status=0 means a successful termination. For other values we recommend to inspect the help page of the selected solver, the name of the solver can be returned by the function getSolver. Use the function setSolver to reset the value to NA if it should be required.

Optim Slot:
The name of the default solver used for optimization can be retrieved calling the function getSolver. The default value for the value solver in the specification is set to NULL which means that the best solver available will be autoselected and used. Before optimization the user can change the setting to another solver. Be aware, that a possible personal change will be overwritten by the function setType, so call setSolver after setting the type of the portfolio.

The logical flag trace in the slot optim allows to trace optionally the portfolio optimization process. By default this will not be the case since the default value is trace=FALSE. Use the function setTrace to modify the selection.

Retrieving and Modifying Specification Settings:
Information about the current portfolio specification can be retrieved by "get" functions. These include:

- `getPortfolioType`: Extracts portfolio type from specification,
- `getOptimize`: Extracts what to optimize from specification,
- `getEstimator`: Extracts type of covariance estimator,
- `getTailRisk`: Extracts list of tail dependency risk matrixes,
- `getParams`: Extracts parameters from specification,
- `getWeights`: Extracts weights from a portfolio object,
- `getTargetReturn`: Extracts target return from specification,
- `getTargetRisk`: Extracts target risks from specification,
- `getAlpha`: Extracts target VaR-alpha specification,
- `getRiskFreeRate`: Extracts risk free rate from specification,
- `getNFrontierPoints`: Extracts number of frontier points,
- `getStatus`: Extracts the status of optimization,
- `getSolver`: Extracts solver from specification,
- `getTrace`: Extracts solver’s trace flag.

For details we refer to `link{getSpec}`.

To modify the setting from a portfolio specification use the "set" functions:

- `setPortfolioType`: Sets type of portfolio optimization,
- `setOptimize`: Sets what to optimize, min risk or max return,
- `setEstimator`: Sets names of mean and covariance estimators,
- `setParams`: Sets optional model parameters,
- `setWeights`: Sets weights vector,
- `setTargetReturn`: Sets target return value,
- `setTargetRisk`: Sets target risk value,
- `setTargetAlpha`: Sets CVaR target alpha value,
- `setRiskFreeRate`: Sets risk-free rate value,
- `setNFrontierPoints`: Sets number of frontier points,
- `setStatus`: Sets status value,
- `setSolver`: Sets the type of solver to be used,
- `setTrace`: Sets the logical trace flag.

For details we refer to `link{setSpec}`.

**Printing Specification Settings:**

There is a generic print function to print information from specification. What is printed depends on the values of the settings. For example `print(portfolioSpec())` returns the type of portfolio, the name of the covariance estimator, the portfolios risk free rate, and the desired solver.

**Value**

`portfolioSpec`

returns an S4 object of class "FPFOLIOSPEC".
References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

portfolio-riskPfolio

Risk and Related Measures for Portfolios

Description

Computes Value-at-Risk and related measures for a portfolio of assets.

The functions are:

- `pfoliovar` computes Value-at-Risk for a portfolio of assets,
- `pfoliocvarplus` computes Value-at-Risk+ for a portfolio of assets,
- `pfoliocvar` computes Conditional Value-at-Risk for a portfolio of assets,
- `lambdacvar` computes CVaR's atomic split value lambda,
- `pfoliomaxloss` computes Maximum Loss for a portfolio of assets,
- `pfolioreturn` computes return values of a portfolio,
- `pfoliotargetreturn` computes the target return of a portfolio,
- `pfoliotargetrisk` computes the target risk of a portfolio,
- `pfoliohist` plots a histogram of the returns of a portfolio.

Usage

```r
pfoliovar(x, weights = NULL, alpha = 0.05)
pfoliocvarplus(x, weights = NULL, alpha = 0.05)
pfoliocvar(x, weights = NULL, alpha = 0.05)
lambdacvar(n, alpha = 0.05)
pfoliomaxloss(x, weights = NULL)
pfolioreturn(x, weights = NULL, geometric = FALSE)
pfoliotargetreturn(x, weights = NULL)
pfoliotargetrisk(x, weights = NULL)
pfoliohist(x, weights = NULL, alpha = 0.05, range = NULL, details = TRUE, ...)
```

Arguments

- `x` a `timeSeries` object, data frame or any other rectangular object which can be expressed as a matrix. The first dimension is the number of observations, we call it `n`, and the second is the number of assets in the data set, we call it `dim`.
- `weights` usually a numeric vector which has the length of the number of assets. The weights measures the normalized weights of the individual assets. By default `NULL`, then an equally weighted set of assets is assumed.
- `geometric` a logical flag, should geometric returns be used, by default `FALSE`.
- `alpha` a numeric value, the confidence interval, by default `0.05`.

details a logical value, should details be printed?

n the number of observation from which the CVaR’s atomic split value \( \lambda = 1 - \text{floor}(\alpha n)/(\alpha n) \) will be evaluated.

range a numeric vector of two elements limiting the plot range of the histogram. This is quite useful if one likes to compare several plots on the same scale. If range=NULL, the default value, then the range will be selected automatically.

... optional arguments to be passet to the function hist.

Details

The percentile measures of loss (or reward) are defined in the following way: Let \( f(x, y) \) be a loss functions depending upon a decision vector \( x = (x_1, \ldots, x_n) \) and a random vector \( y = (y_1, \ldots, y_m) \), then

\[ \text{portfolioVar} \]

is the alpha-percentile of the loss distribution, a smallest value such that the probability that losses exceed or are equal to this value is greater or equal to alpha.

\[ \text{portfolioCVaRplus} \] or "CVaR+" or the "upper CVaR" are the expected losses strictly exceeding VaR. This is also also called "Mean Excess Loss" and "Expected Shortfall".

\[ \text{portfolioCVaR} \]

is a weighted average of VaR and CVaRplus defined as

\[ CVaR = \lambda \ast VaR + (1 - \lambda) CVaRplus, \]

for \( 0 < \lambda < 1 \).

Note, CVaR is convex, but VaR and CVaRplus may be non-convex. The following inequalities are valid: \( VaR < CVaR < CVaRplus \).

Value

\[ \text{portfolioVar} \]

returns the value of risk, VaR, for a portfolio of assets, a numeric value.

\[ \text{portfolioCVaRplus} \]

returns the conditional value of risk plus, CVaRplus, for a portfolio of assets, a numeric value.

\[ \text{portfolioCVaR} \]

returns the conditional value of risk, CVaR, for a portfolio of assets, a numeric value.

\[ \lambda \text{CVaR} \]

returns CVaR’s atomic split value \( \lambda \), a numeric value.

\[ \text{portfolioMaxLoss} \]

returns the maximum loss value of the portfolio, a numeric value.

\[ \text{portfolioReturn} \]

returns the total portfolio return computed from the set of assets \( x \), a numeric vector.

\[ \text{portfolioTargetReturn} \]

returns the total return or target return computed from the set of assets \( x \) and weights \( \text{weights} \), a
numeric value.

\texttt{portfolioTargetRisk}  
returns the total risk (Sigma) or target risk computed from the set of assets $x$ and weights via the formula $\sqrt{\text{weights} \times \text{cov}(x) \times \text{weights}}$, a numeric value.

\texttt{portfolioHist}  
plots a histogram of portfolio returns and adds the values for the VaR (blue), for the CVaRplus (red), and for the maximum loss (green) to the histogram plot. The function invisibly returns a list with the following elements: VaR, VaRplus, maxLoss, mean, and sd. If details is TRUE, then the result is printed.

\textbf{References}  

\textbf{portfolio-rollingPortfolios}  
\textit{Rolling Portfolio}  
\textbf{Description}  
A collection and description of functions allowing to roll a portfolio optimization over time.

The functions are:

\begin{itemize}
  \item \texttt{rollingWindows}  
    Returns a list of rolling window frames,
  \item \texttt{rollingCmlPortfolio}  
    Rolls a CML portfolio,
  \item \texttt{rollingTangencyPortfolio}  
    Rolls a tangency portfolio,
  \item \texttt{rollingMinvariancePortfolio}  
    Rolls a minimum risk portfolio,
  \item \texttt{rollingPortfolioFrontier}  
    returns an efficient portfolio
\end{itemize}

\textbf{Usage}  
\texttt{rollingWindows(x, period = "12m", by = "1m")}

\texttt{rollingCmlPortfolio(data, spec, constraints, from, to, action = NULL,  
  title = NULL, description = NULL, \ldots)}

\texttt{rollingTangencyPortfolio(data, spec, constraints, from, to, action = NULL,  
  title = NULL, description = NULL, \ldots)}

\texttt{rollingMinvariancePortfolio(data, spec, constraints, from, to, action = NULL,  
  title = NULL, description = NULL, \ldots)}

\texttt{rollingPortfolioFrontier(data, spec, constraints, from, to, action = NULL,  
  title = NULL, description = NULL, \ldots)}
title = NULL, description = NULL, ...

Arguments

- **action**: a character string naming a user defined function. This function is optionally applied after each rolling step.
- **by**: a character string, by default "1m", which denotes 1 month. The shift by which the portfolio is rolled.
- **constraints**: a character string vector, containing the constraints of the form "minW[asset]=percentage" for box constraints resp. "maxsumW[assets]=percentage" for sector constraints.
- **data**: a list, having a statistics named list, having named entries 'mu' and 'Sigma', containing the information of the statistics.
- **description**: a character string, allowing for a brief project description, by default NULL, i.e. Date and User.
- **from, to**: a vector of S4 timeDate objects which denote the starting and ending dates for the investigation.
- **period**: a character string, by default "12m", which denotes 12 months. The period over which the portfolio is rolled.
- **spec**: an S4 object of class `fpfoliospec`.
- **title**: a character string, containing the title for the object, by default NULL.
- **x**: an S4 object of class `timeseries` from which the rolling window frames will be created. The length of these frames is given by the argument period and they are shifted by the value specified by the argument by.
- **...**: optional arguments to be passed.

Details

**Rolling Windows**: The function `rollingwindows` constructs from a 'timeSeries' object windows frames of given length period and shift by. ...

**Rolling Portfolios**:

The functions `rolling*portfolio` ...

**Rolling Frontier**:

The function `rollingPortfolioFrontier` ...

Value

`rollingwindows()` returns ...
portfolio-setSpec

rollingCmlPortfolio
rollingTangencyPortfolio
rollingMinvariancePortfolio
return ...

rollingPortfolioFrontier
returns ...

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

portfolio-setSpec  Settings for Specifications of Portfolios

Description

Functions to set specifications for a portfolio.

Usage

setType(spec) <- value
setOptimize(spec) <- value
setEstimator(spec) <- value
setTailRisk(spec) <- value
setParams(spec, name) <- value
setAlpha(spec) <- value

setWeights(spec) <- value
setTargetReturn(spec) <- value
setTargetRisk(spec) <- value
setRiskFreeRate(spec) <- value
setNFrontierPoints(spec) <- value
setStatus(spec) <- value

setSolver(spec) <- value
setObjective(spec) <- value
setTrace(spec) <- value
**Arguments**

- **spec**: an S4 object of class `fPFOLIOSPEC`, the specification to be modified, by default the default of the function `portfolioSpec()`.
- **name**: a character string, the name of the value to be set.
- **value**: a value for that component of spec to be set.

**Details**

- **setType**: Sets type of portfolio optimization,
- **setOptimize**: Sets what to optimize, min risk or max return,
- **setEstimator**: Sets names of mean and covariance estimators,
- **setParams**: Sets optional model parameters,
- **setWeights**: Sets weights vector,
- **setTargetReturn**: Sets target return value,
- **setTargetRisk**: Sets target risk value,
- **setTargetAlpha**: Sets CVaR target alpha value,
- **setRiskFreeRate**: Sets risk-free rate value,
- **setNFrontierPoints**: Sets number of frontier points,
- **setStatus**: Sets status value,
- **setSolver**: Sets the type of solver to be used,
- **setObjective**: Sets objective function name to be used,
- **setTrace**: Sets the logical trace flag.

**Value**

- **setType**
- **setOptimize**
- **setEstimator**
- **setParam**

*Model Settings:* just modify the model settings including the portfolio type, the mean/covariance estimator, and optional parameters of an existing portfolio structure.

- **setWeights**
- **setTargetReturn**
- **setTargetRisk**
- **setTargetAlpha**
- **setRiskFreeRate**
- **setNFrontierPoints**
- **setStatus**

*Portfolio Settings:* just modify the portfolio settings including predefined weights, the target return, the risk free rate, the number of frontier points, and the return and risk range of an existing portfolio structure.
Optim Settings: just modifies the solver setting, i.e. the type of solver to be used for portfolio optimization.

References
Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

---

risk-budgeting  

Description
Functions for risk budgeting.

Usage

```r
sampleCOV(x)
normalVaR(x, alpha=0.05)
modifiedVaR(x, alpha=0.05)
sampleVaR(x, alpha=0.05)
```

```r
budgetsSampleCOV(x, weights, mu=NULL, Sigma=NULL)
budgetsNormalVaR(x, weights, alpha=0.05, mu=NULL, Sigma=NULL)
budgetsModifiedVaR(x, weights, alpha=0.05, mu=NULL, Sigma=NULL, M3=NULL, M4=NULL)
budgetsNormalES(x, weights, alpha=0.05, mu=NULL, Sigma=NULL)
budgetsModifiedES(x, weights, alpha=0.05, mu=NULL, Sigma=NULL, M3=NULL, M4=NULL)
```

Arguments

- `x`  
- `weights`  
- `alpha`  
- `mu, Sigma`  
- `M3, M4`
References


Description

Functions for surface risk analytics.

Usage

markowitzHull(data, nFrontierPoints=50)
feasibleGrid(hull, trace=FALSE)
bestDiversification(grid, FUN="var", trace=FALSE)

riskSurface(diversification, FUN=NULL, ...)

surfacePlot(surface, type=c("image", "filled.contour"), nlevels=11,
palette=topo.colors, addContour=TRUE, addGrid=TRUE, addHull=TRUE,
addAssets=TRUE, ...)

Arguments

data       data
hull       hull
surface    surface
diversification    diversification
FUN       FUN
grid       grid
nFrontierPoints nFrontierPoints
trace      trace
type       type
nlevels    nlevels
palette    palette
addContour addContour
addGrid    addGrid
addHull    addHull
addAssets addAssets
...    optional arguments
risk-ternaryMap

Creates and Plots a Ternary Map

Description

Functions for creating and plotting ternary maps.

Usage

ternaryMap(data, FUN=NULL, ..., locator=FALSE, N=41, palette=topo.colors, nlevels=11)

riskMap(data, locator=FALSE)

ternaryFrontier(data, locator=FALSE)

ternaryWeights(n=21)

ternaryCoord(weights)

ternaryPoints(weights, ...)

Arguments

data data

weights weights

FUN, locator, N, palette, nlevels

ternaryMap

n n

... optional arguments

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
solve-environment  Nonlinear Objective Presettings

Description
Presets variables for Data, portfolioObjective, portfolioReturn, and portfolioRisk in the case of NL math programming of portfolios.

Usage
Data
portfolioObjective(weights)
portfolioReturn(weights)
portfolioRisk(weights)

Arguments
weights a vector of portfolio weights

References
Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

solver-ampl  AMPL Interface

Description
R/AMPL Interface functions.

Usage
amplModelOpen(project)
amplModelAdd(model, project)
amplModelShow(project)
amplDataOpen(project)
amplDataAdd(name, data, type, project)
amplDataAddValue(data, value, project)
amplDataAddVector(data, vector, project)
amplDataAddMatrix(data, matrix, project)
amplDataSemicolon(project)
amplDataShow(project)
amplRunOpen(project)
amplRunAdd(run, project)
amplRunShow(project)
amplOutShow(project)

Arguments

- **project**: a character string, the AMPL project name.
- **model**: ...
- **data**: ...
- **run**: ...
- **type**: ...
- **name**: ...
- **value**: ...
- **vector**: ...
- **matrix**: ...

Value

returns AMPL files.

Author(s)

Diethelm Wuertz.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Usage

solveGlpk.CVAR(data, spec, constraints)
solveGlpk.MAD(data, spec, constraints)
solveRmpl.CVAR(data, spec, constraints)

solveRshortExact(data, spec, constraints)
solveRquadprog(data, spec, constraints)
solveRquadprog.CLA(data, spec, constraints)
solveRipop(data, spec, constraints)
solveRmpl.MV(data, spec, constraints)

solveRsocp(data, spec, constraints)
solveRdonlp2(data, spec, constraints)
solveRsolnp(data, spec, constraints)

Arguments

data a time series or a named list, containing either a series of returns or named entries 'mu' and 'Sigma' being mean and covariance matrix.
spec an S4 object of class fPFOLIOSPEC as returned by the function portfoliospec.
constraints a character string vector, containing the constraints of the form "minW[asset]=percentage" for box constraints resp. "maxsumW[assets]=percentage" for sector constraints.

Value

a list with the following named entries: solver, optim, weights, targetReturn, targetRisk, objective, status, message.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

utils-methods

Print Method for Solvers

Description

S3 print method for mathematical programming solvers.

Usage

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


weights-linePlot

Portfolio Weights Line Plots

Description
Displays line plots of weights, weighted returns, covariance and tail risk budgets.

Usage

weightsLinePlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

weightedReturnsLinePlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

covRiskBudgetsLinePlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

Arguments

object
an S4 object of class fPORTFOLIO, as returned by one of the portfolio functions, e.g. efficientPortfolio or portfolioFrontier.

labels
a logical flag, determining if the graph should be labeled automatically, which is the default case labels=TRUE. If set to FALSE then the graph will be displayed undecorated and the user can decorate it himself.

col
a character string vector, defined from a color palette. The default setting uses the "Blues" seqPalette palette.

title
a logical flag. Should automatically a title and axis labels be added to the plot.

box
a logical flag, determining whether a boxed frame should be plotted around the pie, by default the value is set to TRUE.

References
Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
weights-piePlot

legend

a logical value, determining if the graph should be labeled automatically, which is the default case labels=TRUE. If set to FALSE then the graph will be displayed undecorated and the user can it decorate by himself. Evenmore, if labels takes the value of a string vector, then the names of the assets from the portfolio object will be ignored, and the labels will be taken from the specified string vector.

... additional arguments passed to the function barplot. Only active if labels=FALSE.

Details

These line plots allow for different views on the results obtained from a feasible or an optimized portfolio.

The function weightsPlot displays the weights composition along the frontier of a portfolio.

The function weightedReturnsPlot displays the investment composition, i.e. the weighted returns along the frontier of a portfolio.

The function covRiskBudgetsPlot displays the covariance risk budgets composition along the frontier of a portfolio.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

weights-piePlot  Portfolio Pie Plots

Description

Displays pie plots of weights, weighted Returns, covariance and tail risk budgets for a portfolio.

Usage

weightsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

weightedReturnsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

covRiskBudgetsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

tailRiskBudgetsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

Weights-piePlot

weights-piePlot  Portfolio Pie Plots

Description

Displays pie plots of weights, weighted Returns, covariance and tail risk budgets for a portfolio.

Usage

weightsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

weightedReturnsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

covRiskBudgetsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

tailRiskBudgetsPie(object, pos = NULL, labels = TRUE, col = NULL, box = TRUE, legend = TRUE, radius = 0.8, ...)

Arguments

object  
an S4 object of class fPORTFOLIO, as returned by one of the portfolio functions,  
e.g. efficientPortfolio or portfolioFrontier.

pos  
NULL or an integer value. If NULL it is assumend that we consider a single  
portfolio like for example a tengency portfolio. However, if the object de-  
dcribes a whole frontier then pos has to be the number of that point from the  
frontier which we want to display. The frontier points are numbered from one  
up to the value give by the number of frontier points, which can be retrieved by  
calling getNFrontierPoints.

labels  
a logical flag, determining if the graph should be labeled automatically, which is  
the default case labels=TRUE. If set to FALSE then the graph will be displayed  
undecorated and the user can it decorate by himself. Evenmore, if labels takes  
the value of a string vector, then the names of the assets from the portfolio  
object will be ignored, and the labels will be taken from the specified string  
vector.

col  
a character string vector, defined from a color palette. The default setting uses  
the "Blues" seqPalette palette.

box  
a logical flag, determining whether a boxed frame should be plotted around the  
pie, by default the value is set to TRUE.

legend  
a logical flag, determining if a legend should be added to the plot. The default  
setting shows the legend.

radius  
a numeric value, determining the radius of the pie. The default value is 0.8.

...  
arguments to be passed.

Details

The pie plots allow for different views on the results obtained from a feasible or an optimized  
portfolio.

The function weightsPie displays the weights composition of a portfolio.

The function weightedReturnsPie displays the investment, i.e. the weighted returns of a portfolio.

The function covRiskBudgetsPie displays the covariance risk budgets of a portfolio.

The function tailRiskBudgetsPie displays the copulae tail risk budgets of a portfolio. Note, this  
is only possible if in the portfolio specificsation a copulae tail risk is defined.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics  
eBook, Rmetrics Association and Finance Online, Zurich.
weights-Slider  

_{Portfolio Weights Slider}_

Description

Interactive portfolio weights plot.

Usage

```
weightsSlider(object, control = list(), ...)
```

Arguments

- **control** 
  a list, defining the plotting parameters. The list modifies amongst others the color, e.g. `minvariance.col`, type of point, e.g. `tangency.pch`, or the dimension of the point, e.g. `cml.cex`, see Notes for a complete list of control parameters.

- **object** 
  an S4 object of class `fPORTFOLIO`.

- **...** 
  optional arguments to be passed.

Details

The slider has illustrative objectives. The function expects an S4 object of class `fPORTFOLIO`.

The weights slider gives an overview of the weights on the efficient frontier. Three weight plots `weightsPlot`, `piePlot` and the not stacked weights and a frontier plot with the single assets, the tangency portfolio and a legend are provided. In the two weights plots the vertical line indicates the current portfolio and a dotted one indicates the minimum variance portfolio. The number in the pie plot stands for the asset and the sign shows whether this asset is short or long. In all plots colors represent the same asset.

Value

Creates interactive plots.

Control Parameters

In the following all elements of argument control from functions `plot`, `weightsSlider`, `frontierSlider` are listed.

- **sliderResolution** 
  a numeric, determining the numbers of slider points, by default `nFrontierPoints/10`.

- **sliderFlag** 
  a character string, denoting the slidertype, by default "frontier" for `frontierSlider` and "weights" for `weightsSlider`.

- **sharpeRatio.col** 
  a character string, defining color of the Sharpe ratio plot, by default "black".

- **minvariance.col** 
  a character string, defining color of the minimum variance portfolio, by default "red".
**tangency.col** a character string, defining color of the tangency portfolio, by default "steelblue".

**cml.col** a character string, defining color of the market portfolio and the capital market line, by default "green".

**equalWeights.col** a character string, defining the color of the equal weights portfolio, by default "blue".

**runningPoint.col** a character string, defining color of the point indicating the current portfolio, by default "red".

**singleAsset.col** a character string vector, defining color of the single asset portfolios. The vector must have length the number of assets, by default `rainbow`.

**twoAssets.col** a character string, defining color of the two assets efficient frontier, by default "grey".

**monteCarlo.col** a character string, defining color of the Monte Carlo portfolios, by default "black".

**minvariance.pch** a number, defining symbol used for the minimum variance portfolio. See `points` for description. Default symbol is 17.

**tangency.pch** a number, defining symbol used for the tangency portfolio. See `points` for description. Default symbol is 17.

**cml.pch** a number, defining symbol used for the market portfolio. See `points` for description. Default symbol is 17.

**equalWeights.pch** a number, defining symbol used for the equal weights portfolio. See `points` for description. Default symbol is 15.

**singleAsset.pch** a number, defining symbol used for the single asset portfolios. See `points` for description. Default symbol is 18.

**sharpeRatio.cex** a number, determining size (percentage) of the Sharpe ratio plot, by default 0.1.

**minvariance.cex** a number, determining size (percentage) of the minimum variance portfolio symbol, by default 1.

**tangency.cex** a number, determining size (percentage) of the tangency portfolio symbol, by default 1.25.

**cml.cex** a number, determining size (percentage) of the market portfolio symbol, by default 1.25.

**equalWeights.cex** a number, determining size (percentage) of the equal weights portfolio symbol, by default 0.8.

**runningPoint.cex** a number, determining size (percentage) of the point indicating the current portfolio equal weights portfolio symbol, by default 0.8.

**singleAsset.cex** a number, determining size (percentage) of the single asset portfolio symbols, by default 0.8.

**twoAssets.cex** a number, determining size (percentage) of the two assets efficient frontier plot, by default 0.01.

**monteCarlo.cex** a number, determining size (percentage) of the Monte Carlo portfolio symbols, by default 0.01.

**mcSteps** a number, determining number of Monte Carlo portfolio, by default 5000.
weightsPlot

**pieR**  a vector, containing factors for shrinking and stretching the x- and y-axis, by default NULL, i.e. c(1, 1) is used. Default pie size is 1/15 of the plot range.

**piePos**  a number, determining the weight on the efficient frontier, which is illustrated by the pie. Default is tangency portfolio

**pieOffset**  a vector, containing the pie’s x- and y-axis offset from the efficient frontier. Default is NULL, i.e. the pie is set one default radius left of the efficient frontier.

**xlim**  a vector, containing x-axis plot limits of the efficient frontier. Default setting is maximum of frontier range or single assets portfolios.

**ylim**  a vector, containing y-axis plot limits of the efficient frontier. Default setting is maximum of frontier range or single assets portfolios.

**References**

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

weightsPlot  

*Portfolio Weights Bar Plots*

**Description**

Displays plots of weights, investments, covariance and tail risk budgets.

**Usage**

```r
weightsPlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

weightedReturnsPlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

covRiskBudgetsPlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

tailRiskBudgetsPlot(object, labels = TRUE, col = NULL, title = TRUE, box = TRUE, legend = TRUE, ...)

riskBudgetsPlot(object, FUN=c("budgetsNormalVAR","budgetsNormalES","budgetsModifiedVAR","budgetsModifiedES","budgetsSampleCOV"), labels = TRUE, col = NULL, title = TRUE, mtext = TRUE, box = TRUE, legend = TRUE, ...)
```
Arguments

- **object**: an S4 object of class `fPortfolio`, as returned by one of the portfolio functions, e.g. `efficientPortfolio` or `portfolioFrontier`.
- **labels**: a logical flag, determining if the graph should be labeled automatically, which is the default case `labels=TRUE`. If set to `FALSE` then the graph will be displayed undecorated and the user can it decorate by himself.
- **col**: a character string vector, defined from a color palette. The default setting uses the "Blues" seqPalette palette.
- **title**: a logical flag. Should automatically a title and axis labels be added to the plot.
- **box**: a logical flag, determining whether a boxed frame should be plotted around the pie, by default the value is set to `TRUE`.
- **legend**: a logical value, determining if the graph should be labeled automatically, which is the default case `labels=TRUE`. If set to `FALSE` then the graph will be displayed undecorated and the user can it decorate by himself. Evenmore, if `labels` takes the value of a string vector, then the names of the assets from the portfolio object will be ignored, and the labels will be taken from the specified string vector.
- **...**: additional arguments passed to the function `barplot`. Only active if `labels=FALSE`.
- **FUN**: `FUN`
- **mtext**: `mtext`

Details

These barplots plots allow for different views on the results obtained from a feasible or an optimized portfolio.

The function `weightsPlot` displays the weights composition along the frontier of a portfolio.

The function `weightedReturnsPlot` displays the investment composition, i.e. the weighted returns along the frontier of a portfolio.

The function `covRiskBudgetsPlot` displays the covariance risk budgets composition along the frontier of a portfolio.

The function `tailRiskBudgetsPlot` displays the copulae tail risk budgets composition along the frontier of a portfolio. Note, this is only possible if in the portfolio specifications a copulae tail risk is defined.

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); *Portfolio Optimization with R/Rmetrics*, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.
Index

*Topic **datasets**
data-sets, 13

*Topic **math**
portfolio-riskPfolio, 56

*Topic **models**
backtest-constructors, 3
backtest-extractors, 4
backtest-functions, 6
backtest-getMethods, 7
backtest-performance, 8
backtest-plots, 9
backtest-portfolio, 10
backtest-specification, 11
backtestStats, 12
fPFOLOIBACKTEST, 14
fPFOLIOCON, 15
fPFOLIODATA, 15
fPFOLIOSPEC, 16
fPFOLIOVAL, 18
fPORTFOLIO, 18
fPortfolio-package, 3
frontier-plot, 22
frontier-plotControl, 24
frontier-points, 26
mathprog-LP, 27
mathprog-NLP, 28
mathprog-QP, 31
methods-plot, 33
methods-show, 33
methods-summary, 34
monitor-stability, 34
portfolio-constraints, 35
portfolio-covEstimator, 37
portfolio-dataSets, 39
portfolio-efficientPortfolio, 39
portfolio-feasiblePortfolio, 40
portfolio-getData, 41
portfolio-getDefault, 42
portfolio-getPortfolio, 44
portfolio-getSpec, 46
portfolio-getVal, 48
portfolio-pfolioRisk, 49
portfolio-portfolioFrontier, 49
portfolio-portfolioSpec, 50
portfolio-rollingPortfolios, 58
portfolio-setSpec, 60
risk-budgeting, 62
risk-surfaceRisk, 63
risk-ternaryMap, 64
solver-family, 66
utils-methods, 67
weights-linePlot, 68
weights-piePlot, 69
weights-Slider, 71
weightsPlot, 73

*Topic **optim**
solve-environment, 65
solver-ampl, 65

addRainbow (monitor-stability), 34
amplDataAdd (solver-ampl), 65
amplDataAddMatrix (solver-ampl), 65
amplDataAddValue (solver-ampl), 65
amplDataAddVector (solver-ampl), 65
amplDataOpen (solver-ampl), 65
amplDataSemicolon (solver-ampl), 65
amplDataShow (solver-ampl), 65
amplLP (mathprog-LP), 27
amplLPControl (mathprog-LP), 27
amplModelAdd (solver-ampl), 65
amplModelOpen (solver-ampl), 65
amplModelShow (solver-ampl), 65
amplNLP (mathprog-NLP), 28
amplNLPControl (mathprog-NLP), 28
amplOutShow (solver-ampl), 65
amplQP (mathprog-QP), 31
amplQPControl (mathprog-QP), 31
amplRunAdd (solver-ampl), 65
amplRunOpen (solver-ampl), 65
amplRunShow (solver-ampl), 65
backtest-constructors, 3
backtest-extractors, 4
backtest-functions, 6
backtest-getMethods, 7
backtest-performance, 8
backtest-plots, 9
backtest-portfolio, 10
backtest-specification, 11
backtestAssetsPlot (backtest-plots), 9
backtestDrawdownPlot (backtest-plots), 9
backtestPlot (backtest-plots), 9
backtestPortfolioPlot (backtest-plots), 9
backtestReportRebalancePlot (backtest-plots), 9
backtestReportPlot (backtest-plots), 9
backtestStats, 12
backtestWeightsPlot (backtest-plots), 9
covestimator (portfolio-covestimator), 37
covMcdeEstimator (portfolio-covEstimator), 37
covOGKEstimator (portfolio-covEstimator), 37
covRisk (portfolio-covRisk), 49
covRiskBudgetsLinePlot (weights-linePlot), 68
covRiskBudgetsPie (weights-piePlot), 69
covRiskBudgetsPlot (weightsPlot), 73
cvarRisk (portfolio-covRisk), 49
Data (solve-environment), 65
data-sets, 13
dataSets (data-sets), 13
donlp2NLP (mathprog-NLP), 28
donlp2NLPControl (mathprog-NLP), 28
drawdownsAnalytics (monitor-stability), 34
ECON85 (data-sets), 13
ECON85LONG (data-sets), 13
efficientPortfolio (portfolio-efficientPortfolio), 39
eqsumWConstraints (portfolio-constraints), 35
equalWeightsPoints (frontier-plot), 22
equidistWindows (backtest-functions), 6
feasibleGrid (risk-surfaceRisk), 63
feasiblePortfolio (portfolio-feasiblePortfolio), 40
fPFOLIOBACKTEST, 14
fPFOLIOBACKTEST-class (fPFOLIOBACKTEST), 14
fPFOLIOCON, 15
fPFOLIOCON-class (fPFOLIOCON), 15
fPFOLIODATA, 15
fPFOLIODATA-class (fPFOLIODATA), 15
fPFOLIOSPEC, 16
fPFOLIOSPEC-class (fPFOLIOSPEC), 16
fPFOLIOVAL, 18
fPFOLIOVAL-class (fPFOLIOVAL), 18
fPortfolio (fPortfolio-package), 18
fPortfolio-package, 3
frontier-plot, 22
frontier-plotControl, 24
frontier-points, 26
frontierPlot (frontier-plot), 22
frontierPlotControl (frontier-plotControl), 24
frontierPoints (frontier-points), 26
garchAnalytics (monitor-stability), 34
INDEX

GCCINDEX (data-sets), 13
getA (portfolio-getSpec), 46
geta.fPORTFOLIO
  (portfolio-getPortfolio), 44
getAlpha (portfolio-getDefault), 42
getAlpha.fPFOLIOSPEC
  (portfolio-getSpec), 46
getAlpha.fPFOLIOVAL (portfolio-getVal), 48
geta.fPORTFOLIO
  (portfolio-getPortfolio), 44
getConstraints (portfolio-getDefault), 42
getConstraints.fPORTFOLIO
  (portfolio-getPortfolio), 44
getConstraintsTypes
  (portfolio-getPortfolio), 44
getControl (portfolio-getDefault), 42
getControl.fPFOLIOSPEC
  (portfolio-getSpec), 46
getControl.fPORTFOLIO
  (portfolio-getPortfolio), 44
getCov (portfolio-getDefault), 42
getCov.fPFOLIODATA (portfolio-getData), 41
getCov.fPORTFOLIO
  (portfolio-getPortfolio), 44
getCovRiskBudgets
  (portfolio-getDefault), 42
getCovRiskBudgets.fPFOLIOVAL
  (portfolio-getVal), 48
getCovRiskBudgets.fPORTFOLIO
  (portfolio-getPortfolio), 44
data (portfolio-getDefault), 42
data.fPFOLIODATA
  (portfolio-getData), 41
data.fPORTFOLIO
  (portfolio-getPortfolio), 44
default (portfolio-getDefault), 42
getAddress (portfolio-getDefault), 42
getAddress.fPFOLIODATA
  (portfolio-getData), 41
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getDefault (portfolio-getDefault), 42
getEstimator (portfolio-getDefault), 42
getAddress.fPFOLIODATA
  (portfolio-getData), 41
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getMean (portfolio-getDefault), 42
getAddress.fPFOLIODATA
  (portfolio-getData), 41
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getMessages (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getMessages.fPFOLIOBACKTEST
  (backtest-extractors), 4
getModel (portfolio-getDefault), 42
getAddress.fPORTFOLIO
  (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getMu (portfolio-getDefault), 42
getAddress.fPORTFOLIO
  (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getData), 41
getMu.fPORTFOLIO
  (portfolio-getPortfolio), 44
getNAssets (portfolio-getDefault), 42
getNAssets.fPFOLIODATA
  (portfolio-getData), 41
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getNFrontierPoints
  (portfolio-getDefault), 42
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getAddress.fPFOLIOVAL
  (portfolio-getVal), 48
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getObjective (portfolio-getDefault), 42
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getSpec), 46
getAddress.fPORTFOLIO
  (portfolio-getPortfolio), 44
getOptim (portfolio-getDefault), 42
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getOptim.fPORTFOLIO
  (portfolio-getPortfolio), 44
getOptimize (portfolio-getDefault), 42
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getOptimize.fPORTFOLIO
  (portfolio-getPortfolio), 44
getOptions (portfolio-getDefault), 42
getAddress.fPFOLIOSPEC
  (portfolio-getSpec), 46
getOptions.fPORTFOLIO


INDEX

(portfolio-getPortfolio), 44
getTargetReturn (portfolio-getDefault), 42
getTargetReturn.fPFOLIOSPEC (portfolio-getSpec), 46
getTargetReturn.fPFOLIOVAL (portfolio-getVal), 48
getTargetReturn.fPORTFOLIO (portfolio-getPortfolio), 44
getTargetRisk (portfolio-getDefault), 42
getTargetRisk.fPFOLIOSPEC (portfolio-getSpec), 46
getTargetRisk.fPFOLIOVAL (portfolio-getVal), 48
getTargetRisk.fPORTFOLIO (portfolio-getPortfolio), 44
getTrace (portfolio-getDefault), 42
getTrace.fPFOLIOSPEC (portfolio-getSpec), 46
getTrace.fPORTFOLIO (portfolio-getPortfolio), 44
g getType (portfolio-getDefault), 42
g getType.fPFOLIOSPEC (portfolio-getSpec), 46
g getType.fPORTFOLIO (portfolio-getPortfolio), 44
ggetUnits (portfolio-getDefault), 42
ggetUnits.fPFOLIODATA (portfolio-getData), 41
ggetUnits.fPORTFOLIO (portfolio-getPortfolio), 44
ggetWeights (portfolio-getDefault), 42
ggetWeights.fPFOLIOSPEC (portfolio-getSpec), 46
ggetWeights.fPFOLIOVAL (portfolio-getVal), 48
ggetWeights.fPORTFOLIO (portfolio-getPortfolio), 44
ggetWindows (backtest-getMethods), 7
ggetWindows.fPFOLIOBACKTEST (backtest-extractors), 4
ggetWindowsFun (backtest-getMethods), 7
ggetWindowsFun.fPFOLIOBACKTEST (backtest-extractors), 4
ggetWindowsHorizon (backtest-getMethods), 7
ggetWindowsHorizon.fPFOLIOBACKTEST (backtest-extractors), 4
ggetWindowProps (backtest-getMethods), 7
ggetWindowProps.fPFOLIOBACKTEST (backtest-extractors), 4
ggetWindowProps (backtest-getMethods), 7
ggetWindowProps.fPFOLIOBACKTEST (backtest-extractors), 4
ggetWindowProps (backtest-getMethods), 7
ggetWindowProps.fPFOLIOBACKTEST (backtest-extractors), 4

getWindowsParams (backtest-getMethods), 7
ggetWindowsParams.fPFOLIOBACKTEST (backtest-extractors), 4
glpkLP (mathprog-LP), 27
glpkLPControl (mathprog-LP), 27
ipopQP (mathprog-QP), 31
ipopQPControl (mathprog-QP), 31
kendallEstimator (portfolio-covEstimator), 37
kestrrelQP (mathprog-QP), 37
kestrrelQPControl (mathprog-QP), 31
lambdaCVaR (portfolio-riskPfolio), 56
listFConstraints (portfolio-constraints), 35
lpmEstimator (portfolio-covEstimator), 37
LPP2005 (data-sets), 13
markowitzHull (risk-surfaceRisk), 63
mathprog-LP, 27
mathprog-NLP, 28
mathprog-QP, 31
maxBCConstraints (portfolio-constraints), 35
maxBuyinConstraints (portfolio-constraints), 35
maxCardConstraints (portfolio-constraints), 35
maxddMap (risk-ternaryMap), 64
maxFConstraints (portfolio-constraints), 35
maxratioPortfolio (portfolio-efficientPortfolio), 39
maxreturnPortfolio (portfolio-efficientPortfolio), 39
maxsumWConstraints (portfolio-constraints), 35
maxWConstraints (portfolio-constraints), 35
mcdEstimator (portfolio-covEstimator), 37
methods-plot, 33
methods-show, 33
setWindowsParams <-
   (backtest-constructors), 3
sharpeRatioLines (frontier-plot), 22
show,FPFOLIOBACKTEST-method
   (FPFOLIOBACKTEST), 14
show,FPFOLIOCON-method (FPFOLIOCON), 15
show,FPFOLIODATA-method (FPFOLIODATA), 15
show,FPFOLIOSPEC-method (FPFOLIOSPEC), 16
show,FPFOLIOVAL-method (FPFOLIOVAL), 18
show,FPFOLIO-meth (methods-show), 33
show-methods (methods-show), 33
shrinkEstimator
   (portfolio-covEstimator), 37
singleAssetPoints (frontier-plot), 22
slpmEstimator (portfolio-covEstimator), 37
SMALLCAP (data-sets), 13
soInpNLPM (mathprog-NLP), 28
soInpNLPControl (mathprog-NLP), 28
solve-environment, 65
solver-ampl, 65
solver-family, 66
solveRampCVAR (solver-family), 66
solveRamp.MV (solver-family), 66
solveRdp1p2 (solver-family), 66
solveRglpk.CV (solver-family), 66
solveRglpk.MAD (solver-family), 66
solveRipop (solver-family), 66
solveRquadprog (solver-family), 66
solveRshortExact (solver-family), 66
solveRsocpl (solver-family), 66
solveRsolnp (solver-family), 66
spearmanEstimator
   (portfolio-covEstimator), 37
SPISECTOR (data-sets), 13
stabilityAnalytics (monitor-stability), 34
summary-methods (methods-summary), 34
summary.FPORTFIO (FPORTFIO), 18
surfacePlot (risk-surfaceRisk), 63
SWX (data-sets), 13
symphonyLP (mathprog-LP), 27
symphonyLPControl (mathprog-LP), 27
tailoredFrontierPlot (frontier-plot), 22
tailRiskBudgetsPie (weights-piePlot), 69
tailRiskBudgetsPlot (weightsPlot), 73
tangencyLines (frontier-plot), 22
tangencyPoints (frontier-plot), 22
tangencyPortfolio
   (portfolio-efficientPortfolio), 39
tangencyStrategy (backtest-functions), 6
ternaryCoord (risk-ternaryMap), 64
ternaryFrontier (risk-ternaryMap), 64
ternaryMap (risk-ternaryMap), 64
ternaryPoints (risk-ternaryMap), 64
ternaryWeights (risk-ternaryMap), 64
turnsAnalytics (monitor-stability), 34
twoAssetsLines (frontier-plot), 22
utils-methods, 67
varRisk (portfolio-pfoRisk), 49
waveletSpectrum (monitor-stability), 34
weightedReturnsLinePlot
   (weights-linePlot), 68
weightedReturnsPie (weights-piePlot), 69
weightedReturnsPlot (weightsPlot), 73
weights-linePlot, 68
weights-piePlot, 69
weights-Slider, 71
weightsLinePlot (weights-linePlot), 68
weightsPie (weights-piePlot), 69
weightsPlot, 73
weightsSlider (weights-Slider), 71