

# Package ‘fPortfolio’

September 28, 2009

**Version** 2100.78

**Revision** 4422

**Date** 2009-09-28

**Title** Rmetrics - Portfolio Selection and Optimization

**Author** Rmetrics Core Team, Diethelm Wuertz

**Depends** R (>= 2.7.0), methods, MASS, quadprog, robustbase, timeDate, timeSeries, fBasics, fAssets  
(>= 2100.78), Rglpk

**Suggests** RUnit, tcltk

**Maintainer** Rmetrics Core Team <Rmetrics-core@r-project.org>

**Description** Environment for teaching “Financial Engineering and Computational Finance”

**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.

**LazyLoad** yes

**LazyData** yes

**License** GPL (>= 2)

**URL** <http://www.rmetrics.org>

**Repository** CRAN

**Date/Publication** 2009-09-28 14:10:22

**R topics documented:**

fPortfolio-package	2
covEstimator	3
dataSets	5
efficientPortfolio	6
feasiblePortfolio	7
fFOLIOCON-class	9
fFOLIODATA-class	10
fPFOLIOSPEC-class	11
fPFOLIOVAL-class	13
fPORTFOLIO-class	13
frontierPlot	17
frontierPlotControl	19
frontierPoints	21
getData	23
getDefault	24
getPortfolio	26
getSpec	29
getVal	31
plot-methods	32
portfolioConstraints	32
portfolioData2	34
portfolioFrontier	35
portfolioRisk	36
portfolioRolling	37
portfolioSpec	39
setSpec	44
show-methods	46
solveRglpk	46
solveRquadprog	47
solveRshortExact	48
summary-methods	49
weightsLinePlot	50
weightsPie	51
weightsPlot	53
weightsSlider	55
<b>Index</b>	<b>58</b>

## Description

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

The implemented portfolio models include the traditional mean–variance Markowitz portfolio, robust variants of the Markowitz portfolio, and the mean-CVaR conditional value-at-Risk portfolio.

Optimization is possible by minimizing the risk if the return is specified.

Linear box/group constraints can be specified.

Depending on the model of the portfolio and the constraints a QP (quadratic programming) and a LP (linear programming) solver are provided for optimization

Several kinds of charts can be produced using graphics tools to visualize the results.

## Details

Package: fPortfolio  
Type: Package  
Date: 2009  
License: GPL Version 2 or later  
Copyright: (c) 1999-2008 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.

---

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, ...)  
  
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 `builtin` from package `covRobust`,  
`shrinkEstimator` uses `builtin` 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

Breiman L. (1996); *Bagging Predictors*, Machine Learning 24, 123–140.  
 Ledoit O., Wolf. M. (2003); *Improved Estimation of the Covariance Matrix of Stock Returns with an Application to Portfolio Selection*, Journal of Empirical Finance 10, 503–621.

Schaefer J., Strimmer K. (2005); *A Shrinkage Approach to Large-Scale Covariance Estimation and Implications for Functional Genomics*, Statist. Appl. Genet. Mol. Biol. 4, 32.

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

### Examples

```
## data -
  colnames (SMALLCAP .RET)

## covEstimator -
  covEstimator (SMALLCAP .RET)

## shrinkEstimator -
  # shrinkEstimator (SMALLCAP .RET)
```

---

dataSets

*Assets Data Sets*

---

### Description

Example data sets for portfolio optimization.

### Usage

```
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.

### Examples

```
## SMALLCAP -
  head (SMALLCAP .RET)
```

---

efficientPortfolio *Efficient Portfolios*

---

## Description

Returns efficient portfolios.

## Usage

```
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 <code>timeSeries</code> . If your timeSerie is not a <code>timeSeries</code> object, consult the generic function <code>as.timeSeries</code> to convert your time series.
spec	an S4 object of class <code>FPFOLIOSPEC</code> as returned by the function <code>portfolioSpec</code> .

## 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`.

### Minumum 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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
setTargetReturn(Spec) = mean(colMeans(Data))
Spec

## constraints -
Constraints = "LongOnly"
Constraints

## efficientPortfolio -
efficientPortfolio(Data, Spec, Constraints)

## tangency Portfolio -
tangencyPortfolio(Data, Spec, Constraints)

## minvariancePortfolio -
minvariancePortfolio(Data, Spec, Constraints)
```

---

feasiblePortfolio *Feasible Portfolios*

---

**Description**

Returns properties of a feasible portfolio.

**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. "maxsumW[assets]=percentage" for sector constraints.
data	a multivariate time series described by an S4 object of class <code>timeSeries</code> . If your <code>timeSerie</code> is not a <code>timeSeries</code> object, consult the generic function <code>as.timeSeries</code> to convert your time series.
spec	an S4 object of class <code>fPFOLIOSPEC</code> as returned by the function <code>portfolioSpec</code> .

**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**

`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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
setWeights(Spec) = rep(0.25, times = 4)
Spec

## constraints -
Constraints = "LongOnly"
Constraints

## feasiblePortfolio -
feasiblePortfolio(Data, Spec, Constraints)
```

---

fFOLIOCON-class      *Portfolio Constraints Handling*

---

## Description

Creates a fPFOLIOCON object from string constraints.

## Usage

```
## 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.

## Examples

```
## getClass-  
getClass("fPFOLIOCON")  
  
## getSlots -  
getSlots("fPFOLIOCON")  
  
## data -  
Data = SMALLCAP.RET  
print(head(Data))  
print(class(Data))  
  
## spec -  
Spec = portfolioSpec()  
setTargetReturn(Spec) = mean(Data)  
  
## constraints -  
Constraints = "LongOnly"  
portfolioConstraints(Data, Spec, Constraints)
```

## Description

Creates a fPFOLIODATA object with data set and statistical measures.

## Usage

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

```
## S4 method for signature 'fPFOLIODATA':
show(object)
```

## 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 `series` and the portfolio `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.

**Examples**

```
## ...
```

---

fPFOLIOSPEC-class *Specification of Portfolios*

---

**Description**

Specifies portfolios.

**Usage**

```
## 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=c("MV", "CVaR")` 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=c("mean", "cov")` 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=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=NULL` a numeric vector specifying the portfolio weights.

`targetReturn=NULL` a numeric value specifying the target return. The default value sets the target return.

`targetRisk=NULL` a numeric value specifying the target risk.

`targetAlpha=NULL` a numeric value specifying the target alpha confidence level for CVaR portfolio optimization. The default value sets the target return.

`riskFreeRate=0` a numeric value specifying the risk free rate.

`nFrontierPoints=50` 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=c("quadprog", "Rdonlp2", "lpSolve")` a character string specifying the name of the solver to be used.

`trace=FALSE` 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 "fPFOLIOSPEC".

## References

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

## Examples

```
## spec -
# Show Default Portfolio Specifications:
Spec = portfolioSpec()

## setRiskFreeRate -
# Change Risk Free Rate
```

```
setRiskFreeRate(Spec) = 3
Spec
```

---

fPFOLIOVAL-class    *Values of Portfolio Frontiers*

---

### Description

Specifies portfolio Optimized Values.

### Usage

```
## S4 method for signature 'fPFOLIOVAL':
show(object)
```

### Arguments

object            an S4 object of class fPFOLIOVAL.

### References

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

### Examples

```
##
```

---

fPORTFOLIO-class    *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

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

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

**Arguments**

<code>control</code>	a list, defining the plotting parameters. The list modifies amongst others the color, e.g. <code>minvariance.col</code> , type of point, e.g. <code>tangency.pch</code> , or the dimension of the point, e.g. <code>cml.cex</code> , see Notes for a complete list of control parameters.
<code>which</code>	which of the plots should be displayed? <code>which</code> 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".
<code>object, x</code>	an S4 object of class <code>fPORTFOLIO</code> .
<code>...</code>	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:

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

## Examples

```
## ---
```

---

frontierPlot                      *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)
```

### 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 <code>frontierPlot</code> 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.
...	optional arguments to be passed.

## Details

<code>frontierPlot</code>	Plots efficient frontier,
<code>minvariancePoints</code>	Adds minimum variance point,
<code>cmlPoints</code>	Adds market portfolio,
<code>cmlLines</code>	Adds capital market Line,
<code>tangencyPoints</code>	Adds tangency portfolio point,
<code>tangencyLines</code>	Adds tangency line,
<code>equalWeightsPoints</code>	Adds point of equal weights portfolio,
<code>singleAssetPoints</code>	Adds points of single asset portfolios,
<code>twoAssetsLines</code>	Adds EF for all combinations of two assets,
<code>sharpeRatioLines</code>	Adds Sharpe ratio line,
<code>monteCarloPoints</code>	Adds randomly produced feasible portfolios,
<code>tailoredFrontierPlot</code>	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.

## Examples

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## portfolioFrontier -
Frontier = portfolioFrontier(Data)
Frontier

## frontierPlot -
frontierPlot(Frontier, pch = 19, xlim = c(0, 0.25), ylim = c(0, 0.035))
grid()
abline(h = 0, col = "grey")
abline(v = 0, col = "grey")

## addon -
minvariancePoints(Frontier, pch = 19, col = "red")
tangencyPoints(Frontier, pch = 19, col = "blue")
tangencyLines(Frontier, col = "blue")
equalWeightsPoints(Frontier, pch = 15, col = "grey")
singleAssetPoints(Frontier, pch = 19, cex = 1.5, col = topo.colors(6))
twoAssetsLines(Frontier, lty = 3, col = "grey")
sharpeRatioLines(Frontier, col = "orange", lwd = 2)

## Feasible Portfolios:
frontierPlot(Frontier, col = c("orange", "orange"), pch = 19)
monteCarloPoints(Frontier, mcSteps = 5000, cex = 0.25, pch = 19)
twoAssetsLines(Frontier, lwd = 2, col = "orange")
```

---

frontierPlotControl

*Frontier Plot Control List*

---

## Description

Allows to modify plot settings for the frontier plot.

**Usage**

```

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           Numer 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.

## Examples

```
## --
```

---

frontierPoints	<i>Get Frontier Points</i>
----------------	----------------------------

---

## 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.

**Examples**

```

## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## portfolioFrontier -
Frontier = portfolioFrontier(Data)

## frontierPoints -
x = frontierPoints(Frontier, risk = "VaR", auto = FALSE)
x = frontierPoints(Frontier, risk = "CVaR", auto = FALSE)

```

---

 getData

*Portfolio Data Extractor Functions*


---

### Description

Extracts information from an object of class fPFOLIODATA.

### Usage

```
## 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':
getNames(object)

## 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.

### Details

getData	Extracts data slot,
getSeries	Extracts assets series,
getNAssets	Extracts number of assets,
getNames	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.

## Examples

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

# portfolioData -
data = portfolioData(Data)

# getData -
getData(data)
getSeries(data)
getNAssets(data)
getNames(data)

# getStatistics -
getStatistics(data)
getMean(data)
getCov(data)
getMu(data)
getSigma(data)
getEstimator(data)

# getTailRisk -
getTailRisk(data)
```

---

getDefault

*Extractor Functions*

---

## 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)
getData(object)
getEstimator(object)
getMean(object)
getMu(object)
getNames(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.

**Examples**

```
## getModel -
```

```

getModel(portfolioSpec())

## getType -
getType(portfolioSpec())

```

---

getPortfolio      *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

```

## 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':
getNames(object)

```

```
## 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`.

### References

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

### Examples

```

## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
Spec

## constraints -
Constraints = "LongOnly"
Constraints

## tangencyPortfolio -
tg = tangencyPortfolio(Data, Spec, Constraints)

```

---

`getSpec`*Portfolio Specification Extractor Functions*

---

**Description**

Extracts information from an object of class `fPFOLIOSPEC`.

**Usage**

```
## S3 method for class 'fPFOLIOSPEC':  
getModel(object)  
## S3 method for class 'fPFOLIOSPEC':  
getType(object)  
## S3 method for class 'fPFOLIOSPEC':  
getOptimize(object)  
## S3 method for class 'fPFOLIOSPEC':  
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

getType	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.

## Examples

```
## spec -
Spec = portfolioSpec()
Spec

# getModel -
getModel(Spec)
getType(Spec)
getOptimize(Spec)
getEstimator(Spec)
getTailRisk(Spec)
```

```

getParams (Spec)

# getPortfolio -
getPortfolio (Spec)
getWeights (Spec)
getTargetReturn (Spec)
getTargetRisk (Spec)
getAlpha (Spec)
getRiskFreeRate (Spec)
getNFrontierPoints (Spec)
getStatus (Spec)

# getOptim -
getOptim (Spec)
getSolver (Spec)
getObjective (Spec)
getOptions (Spec)
getControl (Spec)
getTrace (Spec)

```

---

getVal

*PortfolioVal Extractor Functions*


---

### Description

Extracts information from an object of class fPFOLIOVAL.

### Usage

```

## S3 method for class 'fPFOLIOVAL':
getAlpha(object)
## S3 method for class 'fPFOLIOVAL':
getCovRiskBudgets(object)
## S3 method for class 'fPFOLIOVAL':
getNFrontierPoints(object)
## S3 method for class 'fPFOLIOVAL':
getPortfolio(object)
## S3 method for class 'fPFOLIOVAL':
getRiskFreeRate(object)
## S3 method for class 'fPFOLIOVAL':
getStatus(object)
## S3 method for class 'fPFOLIOVAL':
getTargetReturn(object)
## S3 method for class 'fPFOLIOVAL':
getTargetRisk(object)
## S3 method for class 'fPFOLIOVAL':
getWeights(object)

```

**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.

**Examples**

```
##
```

---

```
plot-methods            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.

---

```
portfolioConstraints
                          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")

listFConstraints(data, spec = portfolioSpec(), constraints = "LongOnly")
minFConstraints(data, spec = portfolioSpec(), constraints = "LongOnly")
maxFConstraints(data, spec = portfolioSpec(), constraints = "LongOnly")

```

## 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 `fPFOLIOSPEC` as returned by the function `portfolioSpec`.

**...** arguments passed to the function `.setRdonlp2Constraints`. 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  $-\text{Inf}$  and  $\text{Inf}$ . 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 and 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** `"minsumW[Asset (s)]=Value (s) "`, and/or  
`"maxsumW[Asset (s)]=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** "minB[Asset (s)]=Value (s) ", and/or  
"minB[Asset (s)]=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: listF, minF, 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.

### **Examples**

```
## constraints -
Constraints <- "LongOnly"
```

---

portfolioData2      *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.

---

portfolioFrontier *Efficient Portfolio Frontier*

---

## Description

Computes the efficient portfolio frontier.

## Usage

```
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 <code>timeSeries</code> . If your <code>timeSerie</code> is not a <code>timeSeries</code> object, consult the generic function <code>as.timeSeries</code> 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 <code>fPFOLIOSPEC</code> as returned by the function <code>portfolioSpec</code> .
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

`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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
Spec
setNFrontierPoints(Spec) = 10

## constraints -
Constraints = "LongOnly"

## portfolioFrontier -
portfolioFrontier(Data, Spec, Constraints)
```

---

 portfolioRisk

*portfolioRisk*


---

**Description**

Computes portfolio risk.

**Usage**

```
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 <code>timeSeries</code> .
weights	a numeric vector of weights.
alpha	a numeric value, the confidence level, by default <code>alpha=0.05</code> , 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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## weights -
nAssets = getNAssets(portfolioData(Data))
Weights <- rep(1/nAssets, times = nAssets)

## covRisk -
covRisk(Data, Weights)

## varRisk -
varRisk(Data, Weights, alpha = 0.05)

## cvarRisk -
cvarRisk(Data, Weights, alpha = 0.05)
```

---

portfolioRolling    *Rolling Portfolio*

---

**Description**

A collection and description of functions allowing to roll a portfolio optimization over time.

The functions are:

rollingWindows	Returns a list of rolling window frames,
rollingCmlPortfolio	Rolls a CML portfolio,
rollingTangencyPortfolio	Rolls a tangency portfolio,
rollingMinvariancePortfolio	Rolls a minimum risk portfolio,
rollingPortfolioFrontier	returns an efficient portfolio

**Usage**

```
rollingWindows(x, period = "12m", by = "1m")

rollingCmlPortfolio(data, spec, constraints, from, to, action = NULL,
  title = NULL, description = NULL, ...)
rollingTangencyPortfolio(data, spec, constraints, from, to, action = NULL,
  title = NULL, description = NULL, ...)
rollingMinvariancePortfolio(data, spec, constraints, from, to, action = NULL,
  title = NULL, description = NULL, ...)

rollingPortfolioFrontier(data, spec, constraints, from, to, action = NULL,
  title = NULL, description = NULL, ...)
```

**Arguments**

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

**Details**

**RollingWindows:** 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 ...

`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.

## Examples

```
## ...
```

---

portfolioSpec	<i>Specification of Portfolios</i>
---------------	------------------------------------

---

## Description

Specifies a portfolio from scratch.

## Usage

```
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 = NULL,
    options = list(meq = 2), control = list(), trace = FALSE),
  messages = list())
```

## 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 <code>solver</code> denoting the type of the solver to be used, a <code>params</code> list to pass further arguments to the objective function to optimize, a <code>control</code> list for all control settings of the solver, and a logical flag, <code>trace</code> denoting if the optimization should be traced.
messages	a list, for optional messages.

## Details

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

```
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:

```
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 `setType` to modify this selection.

#### *What to optimize?*

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

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

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

```
\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 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 `setType` 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:

getType	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:

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,
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.

## Examples

```
## portfolioSpec -
spec = portfolioSpec()

## getRiskFreeRate -
getRiskFreeRate(spec)
spec

## setRiskFreeRate -
setRiskFreeRate(spec) <- 2.5
```

---

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) <- 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 <code>FPFOLIOSPEC</code> , the specification to be modified, by default the default of the function <code>portfolioSpec()</code> .
value	a value for that component of <code>spec</code> 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.

```
setSolver
setObjective
setTrace
```

*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.

**Examples**

```
## portfolioSpec -
# Show Default Portfolio Specifications:
Spec = portfolioSpec()

## setRiskFreeRate -
# Change Risk Free Rate
setRiskFreeRate(Spec) = 3
Spec
```

---

show-methods                      *Portfolio Print Methods*

---

**Description**

show-methods.

**Usage**

```
## 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.

---

solveRglpk                      *Linear Programming Solver*

---

**Description**

Optimizes a portfolio using the linear programming solver Rglpk.

**Usage**

```
solveRglpk(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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
setType(Spec) = "CVaR"
setSolver(Spec) = "solveRglpk"
setTargetReturn(Spec) = mean(Data)
Spec

## constraints -
Constraints = "LongOnly"

## solveRglpk -
solveRglpk(Data, Spec, Constraints)
```

---

solveRquadprog      *Quadratic Programming Solver*

---

**Description**

Optimizes a portfolio with a the quadratic programming solver `quadprog`.

**Usage**

```
solveRquadprog(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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
setSolver(Spec) = "solveRquadprog"
setTargetReturn(Spec) = mean(Data)
Spec

## constraints -
Constraints = "LongOnly"

## solveRquadprog -
solveRquadprog(Data, Spec, Constraints)
```

---

`solveRshortExact` *Exact unlimited Short Selling Solver*

---

**Description**

Optimizes an unlimited short selling portfolio analytically.

**Usage**

```
solveRshortExact(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.

**Examples**

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## spec -
Spec = portfolioSpec()
setSolver(Spec) = "solveRshortExact"
setTargetReturn(Spec) = mean(Data)
Spec

## constraints -
Constraints = "LongOnly"

## solveRshortExact -
solveRshortExact(Data, Spec, Constraints)
```

---

summary-methods      *summary-methods*

---

**Description**

summary-methods.

**References**

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

---

 weightsLinePlot      *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 <code>fPORTFOLIO</code> , as returned by one of the portfolio functions, e.g. <code>efficientPortfolio</code> or <code>portfolioFrontier</code> .
labels	a logical flag, determining if the the graph should be labeled automatically, which is the default case <code>labels=TRUE</code> . If set to <code>FALSE</code> 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" <code>seqPalette</code> 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 <code>TRUE</code> .
legend	a logical value, determining if the the graph should be labeled automatically, shich is the default case <code>labels=TRUE</code> . If set to <code>FALSE</code> then the graph will be displayed undecorated and the user can it decorate by himself. Evenmore, if <code>labels</code> 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 <code>barplot</code> . Only active if <code>labels=FALSE</code> .

**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.

## Examples

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]
Data

## portfolioFrontier -
Frontier = portfolioFrontier(Data)

## weightsLinePlot -
# weightsLinePlot(frontier)
```

---

weightsPie

*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 assumed that we consider a single portfolio like for example a tangency portfolio. However, if the object describes 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 given by the number of frontier points, which can be retrieved by calling <code>getNFrontierPoints</code> .
labels	a logical flag, determining if the graph should be labeled automatically, which is the default case <code>labels=TRUE</code> . If set to <code>FALSE</code> then the graph will be displayed undecorated and the user can decorate it by himself. Even more, if <code>labels</code> 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" <code>seqPalette</code> palette.
box	a logical flag, determining whether a boxed frame should be plotted around the pie, by default the value is set to <code>TRUE</code> .
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 specification 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.

### Examples

```
## data -
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]

## spec -
Spec = portfolioSpec()
setNFrontierPoints(Spec) = 10
```

```

## tangencyPortfolio -
  tg = tangencyPortfolio(Data, Spec)

## weightsPie -
  # par(mfrow = c(2, 2))
  weightsPie(tg)
  weightedReturnsPie(tg)
  covRiskBudgetsPie(tg)

## portfolioFrontier -
  frontier = portfolioFrontier(Data, Spec)

## weightsPie -
  # par(mfrow = c(2, 2))
  weightsPie(frontier, pos = 7)
  weightedReturnsPie(frontier, pos = 7)
  covRiskBudgetsPie(frontier, pos = 7)

```

---

weightsPlot

*Portfolio Weights Pie Plots*


---

### Description

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

### Usage

```

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

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

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

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

```

### Arguments

object	an S4 object of class <code>fPORTFOLIO</code> , as returned by one of the portfolio functions, e.g. <code>efficientPortfolio</code> or <code>portfolioFrontier</code> .
labels	a logical flag, determining if the the graph should be labeled automatically, which is the default case <code>labels=TRUE</code> . If set to <code>FALSE</code> 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" <code>seqPalette</code> palette.

title	a logical flag. Should automatically a title and axis labels be added to the plot.
mtext	a logical flag. Should automatically a margin text added to the right hand side 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 the graph should be labeled automatically, shich 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 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 specifisation 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.

### Examples

```
## data -
  Data = SMALLCAP.RET
  Data = Data[, c("BKE", "GG", "GYMB", "KRON")]

## portfolioFrontier -
  Frontier = portfolioFrontier(Data)

## weightsPlot -
  weightsPlot(Frontier)
```

---

weightsSlider      *Portfolio Weights Slider*


---

**Description**

Interactive portfolio weights plot.

**Usage**

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

**Arguments**

<code>control</code>	a list, defining the plotting parameters. The list modifies amongst others the color, e.g. <code>minvariance.col</code> , type of point, e.g. <code>tangency.pch</code> , or the dimension of the point, e.g. <code>cml.cex</code> , see Notes for a complete list of control parameters.
<code>object</code>	an S4 object of class <code>fPORTFOLIO</code> .
<code>...</code>	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.
- 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.
- 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.

## Examples

```
## Load Data and Convert to timeSeries Object:
Data = SMALLCAP.RET
Data = Data[, c("BKE", "GG", "GYMB", "KRON")]

## portfolioFrontier -
frontier = portfolioFrontier(Data)
frontier

## weightsSlider -
# Try Frontier Slider:
# weightsSlider(frontier)
```

# Index

## \*Topic **models**

- covEstimator, 2
  - dataSets, 4
  - efficientPortfolio, 5
  - feasiblePortfolio, 7
  - fFOLIOCON-class, 8
  - fFOLIODATA-class, 9
  - fPFOLIOSPEC-class, 10
  - fPFOLIOVAL-class, 12
  - fPORTFOLIO-class, 13
  - fPortfolio-package, 2
  - frontierPlot, 16
  - frontierPlotControl, 19
  - frontierPoints, 21
  - getData, 22
  - getDefault, 24
  - getPortfolio, 25
  - getSpec, 28
  - getVal, 30
  - plot-methods, 31
  - portfolioConstraints, 31
  - portfolioData2, 34
  - portfolioFrontier, 34
  - portfolioRisk, 35
  - portfolioRolling, 36
  - portfolioSpec, 38
  - setSpec, 43
  - show-methods, 45
  - solveRglpk, 46
  - solveRquadprog, 47
  - solveRshortExact, 48
  - summary-methods, 49
  - weightsLinePlot, 49
  - weightsPie, 51
  - weightsPlot, 52
  - weightsSlider, 54
- class-fPFOLIOCON  
(*fFOLIOCON-class*), 8
- class-fPFOLIODATA  
(*fFOLIODATA-class*), 9
- class-fPFOLIOSPEC  
(*fPFOLIOSPEC-class*), 10
- class-fPFOLIOVAL  
(*fPFOLIOVAL-class*), 12
- class-fPORTFOLIO  
(*fPORTFOLIO-class*), 13
- cmlLines (*frontierPlot*), 16
- cmlPoints (*frontierPlot*), 16
- covEstimator, 2
- covMcdEstimator (*covEstimator*), 2
- covOGKEstimator (*covEstimator*), 2
- covRisk (*portfolioRisk*), 35
- covRiskBudgetsLinePlot  
(*weightsLinePlot*), 49
- covRiskBudgetsPie (*weightsPie*), 51
- covRiskBudgetsPlot (*weightsPlot*),  
52
- cvarRisk (*portfolioRisk*), 35
- dataSets, 4
- efficientPortfolio, 5
- eqsumWConstraints  
(*portfolioConstraints*), 31
- equalWeightsPoints  
(*frontierPlot*), 16
- feasiblePortfolio, 7
- fFOLIOCON-class, 8
- fFOLIODATA-class, 9
- fPFOLIOCON (*fFOLIOCON-class*), 8
- fPFOLIOCON-class  
(*fFOLIOCON-class*), 8
- fPFOLIODATA (*fFOLIODATA-class*), 9
- fPFOLIODATA-class  
(*fFOLIODATA-class*), 9
- fPFOLIOSPEC (*fPFOLIOSPEC-class*),  
10

- fPFOLIOSPEC-class, 10
- fPFOLIOVAL (*fPFOLIOVAL-class*), 12
- fPFOLIOVAL-class, 12
- fPORTFOLIO (*fPORTFOLIO-class*), 13
- fPortfolio (*fPortfolio-package*), 2
- fPORTFOLIO-class, 13
- fPortfolio-package, 2
- frontierPlot, 16
- frontierPlotControl, 19
- frontierPoints, 21
- GCCINDEX (*dataSets*), 4
- getA (*getSpec*), 28
- getA.fPORTFOLIO (*getPortfolio*), 25
- getAlpha (*getDefault*), 24
- getAlpha.fPFOLIOSPEC (*getSpec*), 28
- getAlpha.fPFOLIOVAL (*getVal*), 30
- getAlpha.fPORTFOLIO (*getPortfolio*), 25
- getConstraints (*getDefault*), 24
- getConstraints.fPORTFOLIO (*getPortfolio*), 25
- getConstraintsTypes (*getPortfolio*), 25
- getControl (*getDefault*), 24
- getControl.fPFOLIOSPEC (*getSpec*), 28
- getControl.fPORTFOLIO (*getPortfolio*), 25
- getCov (*getDefault*), 24
- getCov.fPFOLIODATA (*getData*), 22
- getCov.fPORTFOLIO (*getPortfolio*), 25
- getCovRiskBudgets (*getDefault*), 24
- getCovRiskBudgets.fPFOLIOVAL (*getVal*), 30
- getCovRiskBudgets.fPORTFOLIO (*getPortfolio*), 25
- getData, 22
- getData (*getDefault*), 24
- getData.fPFOLIODATA (*getData*), 22
- getData.fPORTFOLIO (*getPortfolio*), 25
- getDefault, 24
- getEstimator (*getDefault*), 24
- getEstimator.fPFOLIODATA (*getData*), 22
- getEstimator.fPFOLIOSPEC (*getSpec*), 28
- getEstimator.fPORTFOLIO (*getPortfolio*), 25
- getMean (*getDefault*), 24
- getMean.fPFOLIODATA (*getData*), 22
- getMean.fPORTFOLIO (*getPortfolio*), 25
- getMessages (*getSpec*), 28
- getModel.fPFOLIOSPEC (*getSpec*), 28
- getModel.fPORTFOLIO (*getPortfolio*), 25
- getMu (*getDefault*), 24
- getMu.fPFOLIODATA (*getData*), 22
- getMu.fPORTFOLIO (*getPortfolio*), 25
- getNames (*getDefault*), 24
- getNames.fPFOLIODATA (*getData*), 22
- getNames.fPORTFOLIO (*getPortfolio*), 25
- getNAssets (*getDefault*), 24
- getNAssets.fPFOLIODATA (*getData*), 22
- getNAssets.fPORTFOLIO (*getPortfolio*), 25
- getNFrontierPoints (*getDefault*), 24
- getNFrontierPoints.fPFOLIOSPEC (*getSpec*), 28
- getNFrontierPoints.fPFOLIOVAL (*getVal*), 30
- getNFrontierPoints.fPORTFOLIO (*getPortfolio*), 25
- getObjective (*getDefault*), 24
- getObjective.fPFOLIOSPEC (*getSpec*), 28
- getObjective.fPORTFOLIO (*getPortfolio*), 25
- getOptim (*getDefault*), 24
- getOptim.fPFOLIOSPEC (*getSpec*), 28
- getOptim.fPORTFOLIO (*getPortfolio*), 25
- getOptimize (*getDefault*), 24
- getOptimize.fPFOLIOSPEC (*getSpec*), 28
- getOptimize.fPORTFOLIO (*getPortfolio*), 25
- getOptions (*getDefault*), 24
- getOptions.fPFOLIOSPEC (*getSpec*), 28

- getOptions.fPORTFOLIO  
(*getPortfolio*), 25
- getParams(*getDefault*), 24
- getParams.fPFOLIOSPEC(*getSpec*),  
28
- getParams.fPORTFOLIO  
(*getPortfolio*), 25
- getPortfolio, 25
- getPortfolio(*getDefault*), 24
- getPortfolio.fPFOLIOSPEC  
(*getSpec*), 28
- getPortfolio.fPFOLIOVAL(*getVal*),  
30
- getPortfolio.fPORTFOLIO  
(*getPortfolio*), 25
- getRiskFreeRate(*getDefault*), 24
- getRiskFreeRate.fPFOLIOSPEC  
(*getSpec*), 28
- getRiskFreeRate.fPFOLIOVAL  
(*getVal*), 30
- getRiskFreeRate.fPORTFOLIO  
(*getPortfolio*), 25
- getSeries(*getDefault*), 24
- getSeries.fPFOLIODATA(*getData*),  
22
- getSeries.fPORTFOLIO  
(*getPortfolio*), 25
- getSigma(*getDefault*), 24
- getSigma.fPFOLIODATA(*getData*), 22
- getSigma.fPORTFOLIO  
(*getPortfolio*), 25
- getSolver(*getDefault*), 24
- getSolver.fPFOLIOSPEC(*getSpec*),  
28
- getSolver.fPORTFOLIO  
(*getPortfolio*), 25
- getSpec, 28
- getSpec(*getDefault*), 24
- getSpec.fPORTFOLIO  
(*getPortfolio*), 25
- getStatistics(*getDefault*), 24
- getStatistics.fPFOLIODATA  
(*getData*), 22
- getStatistics.fPORTFOLIO  
(*getPortfolio*), 25
- getStatus(*getDefault*), 24
- getStatus.fPFOLIOSPEC(*getSpec*),  
28
- getStatus.fPFOLIOVAL(*getVal*), 30
- getStatus.fPORTFOLIO  
(*getPortfolio*), 25
- getTailRisk(*getDefault*), 24
- getTailRisk.fPFOLIODATA  
(*getData*), 22
- getTailRisk.fPFOLIOSPEC  
(*getSpec*), 28
- getTailRisk.fPORTFOLIO  
(*getPortfolio*), 25
- getTailRiskBudgets(*getDefault*),  
24
- getTailRiskBudgets.fPORTFOLIO  
(*getPortfolio*), 25
- getTargetReturn(*getDefault*), 24
- getTargetReturn.fPFOLIOSPEC  
(*getSpec*), 28
- getTargetReturn.fPFOLIOVAL  
(*getVal*), 30
- getTargetReturn.fPORTFOLIO  
(*getPortfolio*), 25
- getTargetRisk(*getDefault*), 24
- getTargetRisk.fPFOLIOSPEC  
(*getSpec*), 28
- getTargetRisk.fPFOLIOVAL  
(*getVal*), 30
- getTargetRisk.fPORTFOLIO  
(*getPortfolio*), 25
- getTrace(*getDefault*), 24
- getTrace.fPFOLIOSPEC(*getSpec*), 28
- getTrace.fPORTFOLIO  
(*getPortfolio*), 25
- getType(*getDefault*), 24
- getType.fPFOLIOSPEC(*getSpec*), 28
- getType.fPORTFOLIO  
(*getPortfolio*), 25
- getVal, 30
- getWeights(*getDefault*), 24
- getWeights.fPFOLIOSPEC(*getSpec*),  
28
- getWeights.fPFOLIOVAL(*getVal*), 30
- getWeights.fPORTFOLIO  
(*getPortfolio*), 25
- kendallEstimator(*covEstimator*), 2
- listFConstraints  
(*portfolioConstraints*), 31
- lpmEstimator(*covEstimator*), 2

- LPP2005 (*dataSets*), 4
- LPP2005.RET.DF (*dataSets*), 4
- maxBConstraints  
(*portfolioConstraints*), 31
- maxFConstraints  
(*portfolioConstraints*), 31
- maxratioPortfolio  
(*efficientPortfolio*), 5
- maxreturnPortfolio  
(*efficientPortfolio*), 5
- maxsumWConstraints  
(*portfolioConstraints*), 31
- maxWConstraints  
(*portfolioConstraints*), 31
- mcdEstimator (*covEstimator*), 2
- minBConstraints  
(*portfolioConstraints*), 31
- minFConstraints  
(*portfolioConstraints*), 31
- minriskPortfolio  
(*efficientPortfolio*), 5
- minsumWConstraints  
(*portfolioConstraints*), 31
- minvariancePoints (*frontierPlot*),  
16
- minvariancePortfolio  
(*efficientPortfolio*), 5
- minWConstraints  
(*portfolioConstraints*), 31
- monteCarloPoints (*frontierPlot*),  
16
- mveEstimator (*covEstimator*), 2
- nnveEstimator (*covEstimator*), 2
- plot-methods, 31
- plot.fPORTFOLIO  
(*fPORTFOLIO-class*), 13
- points, 15, 55, 56
- portfolioConstraints, 31
- portfolioData (*FFOLIODATA-class*),  
9
- portfolioData2, 34
- portfolioFrontier, 34
- portfolioRisk, 35
- portfolioRolling, 36
- portfolioSpec, 38
- rollingCmlPortfolio  
(*portfolioRolling*), 36
- rollingMinvariancePortfolio  
(*portfolioRolling*), 36
- rollingPortfolio  
(*portfolioRolling*), 36
- rollingPortfolioFrontier  
(*portfolioRolling*), 36
- rollingTangencyPortfolio  
(*portfolioRolling*), 36
- rollingWindows  
(*portfolioRolling*), 36
- setAlpha<- (*setSpec*), 43
- setEstimator<- (*setSpec*), 43
- setNFrontierPoints<- (*setSpec*), 43
- setObjective<- (*setSpec*), 43
- setOptimize<- (*setSpec*), 43
- setParams<- (*setSpec*), 43
- setRiskFreeRate<- (*setSpec*), 43
- setSolver<- (*setSpec*), 43
- setSpec, 43
- setStatus<- (*setSpec*), 43
- setTailRisk<- (*setSpec*), 43
- setTargetReturn<- (*setSpec*), 43
- setTargetRisk<- (*setSpec*), 43
- setTrace<- (*setSpec*), 43
- setType<- (*setSpec*), 43
- setWeights<- (*setSpec*), 43
- sharpeRatioLines (*frontierPlot*),  
16
- show, fPFOLIOCON-method  
(*FFOLIOCON-class*), 8
- show, fPFOLIODATA-method  
(*FFOLIODATA-class*), 9
- show, fPFOLIOSPEC-method  
(*fPFOLIOSPEC-class*), 10
- show, fPFOLIOVAL-method  
(*fPFOLIOVAL-class*), 12
- show, fPORTFOLIO-method  
(*show-methods*), 45
- show-methods, 45
- shrinkEstimator (*covEstimator*), 2
- singleAssetPoints (*frontierPlot*),  
16
- SMALLCAP (*dataSets*), 4
- SMALLCAP.RET.DF (*dataSets*), 4
- solveRglpk, 46
- solveRquadprog, 47

`solveRshortExact`, 48  
`spearmanEstimator` (`covEstimator`),  
2  
`SPISECTOR` (`dataSets`), 4  
summary-methods, 49  
`summary.fPORTFOLIO`  
(`fPORTFOLIO`-class), 13  
`SWX` (`dataSets`), 4  
  
`tailoredFrontierPlot`  
(`frontierPlot`), 16  
`tailRiskBudgetsPie` (`weightsPie`),  
51  
`tailRiskBudgetsPlot`  
(`weightsPlot`), 52  
`tangencyLines` (`frontierPlot`), 16  
`tangencyPoints` (`frontierPlot`), 16  
`tangencyPortfolio`  
(`efficientPortfolio`), 5  
`twoAssetsLines` (`frontierPlot`), 16  
  
`varRisk` (`portfolioRisk`), 35  
  
`weightedReturnsLinePlot`  
(`weightsLinePlot`), 49  
`weightedReturnsPie` (`weightsPie`),  
51  
`weightedReturnsPlot`  
(`weightsPlot`), 52  
`weightsLinePlot`, 49  
`weightsPie`, 51  
`weightsPlot`, 52  
`weightsSlider`, 54