Package 'fPortfolio'

April 25, 2023

Title Rmetrics - Portfolio Selection and Optimization

Date 2023-04-22

Version 4023.84

Description
A collection of functions to optimize portfolios and to analyze them from different points of view.

Depends R (>= 2.15.1), timeDate, timeSeries, fBasics, fAssets

Imports fCopulae, robustbase, MASS, Rglpk, slam, Rsolnp, quadprog, kernlab, rneos, methods, grDevices, graphics, stats, utils

Suggests parma, Rsymphony, dplR, bcp, fGarch, mvoutlier

Additional_repositories https://r-forge.r-project.org/

LazyData yes

License GPL (>= 2)

URL https://r-forge.r-project.org/projects/rmetrics/

NeedsCompilation no

Author Diethelm Wuertz [aut],
Tobias Setz [aut],
Yohan Chalabi [aut],
William Chen [ctb],
Stefan Theussl [aut, cre]

Maintainer Stefan Theussl <Stefan.Theussl@R-project.org>

Repository CRAN

Date/Publication 2023-04-25 07:50:06 UTC

R topics documented:

`fPortfolio-package` *Portfolio Design, Optimization and Backtesting*

Description

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

Details

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

`backtest-constructors` *Specification of backtesting portfolios*

Description

Functions to set specifications for portfolio backtesting.

The functions are:

`setWindowsFun`	Sets Windows function,
`setWindowsParams`	Sets additional parameters for rolling windows function,
`setWindowsHorizon`	Sets Windows horizon,
`setStrategyFun`	Sets the portfolio Strategy function,
`setStrategyParams`	Sets additional parameters for Strategy function,
`setSmootherFun`	Sets the Smoother function,

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

Usage

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

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

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

Arguments

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

Details

The function portfolioBacktest() allows to set the values for the specification structure from scratch.

To modify individual settings one can use the set functions.

References

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

---

backtest-extractors *Portfolio backtest specification extractors*

Description

Extracts information from an object of class fPFOLIOBACKTEST.

The functions are:

getWindows	Extract windows slot,
getWindowsFun	extract windows function,
getWindowsParams	extract a list of windows specific parameters,
getWindowsHorizon	extract windows horizon,
getStrategy	extract strategy slot,
getStrategyFun	extract the portfolio strategy function,
getStrategyParams	extract a list of portfolio strategy specific parameters,
getSmoother	extract the smoother slot,
getSmootherFun	Extract the Ssoother function,
getSmootherParams	extract a list of Smoothing specific parameters,
getSmootherLambda	extract the smoothing parameter Lambda,
getSmootherDoubleSmoothing	extract setting for double smoothing,
getSmootherInitialWeights	extract the initial weights to used in the smoothing,
getSmootherSkip	extract the number of skipped months,
getMessages	extract the message slot.

Usage

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

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

## S3 method for class 'fPFOLIOBACKTEST'
getSmoother(object)
## S3 method for class 'fPFOLIOBACKTEST'
getSmootherFun(object)
## S3 method for class 'fPFOLIOBACKTEST'
getSmootherParams(object)
## S3 method for class 'fPFOLIOBACKTEST'
getSmootherLambda(object)
## S3 method for class 'fPFOLIOBACKTEST'
getSmootherDoubleSmoothing(object)
## S3 method for class 'fPFOLIOBACKTEST'
getSmootherInitialWeights(object)
## S3 method for class 'fPFOLIOBACKTEST'
getSmootherSkip(object)
```

```
## S3 method for class 'fPFOLIOBACKTEST'
getMessages(object)
```

Arguments

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

References

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

---

backtest-functions *User defined functions to perform portfolio backtesting*

Description

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

Usage

```
equidistWindows(data, backtest = portfolioBacktest())

tangencyStrategy(data, spec = portfolioSpec(), constraints = "LongOnly",
    backtest = portfolioBacktest())

emaSmoother(weights, spec, backtest)
```

Arguments

data	a multivariate time series described by an S4 object of class timeSeries. If your timeSerie is not a timeSeries object, consult the generic function as.timeSeries to convert your time series.
backtest	an S4 object of class fPFOLIOBACKTEST as returned by the function portfolioBacktest.
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.
weights	a numeric vector, containing the portfolio weights of an asset

Details

equidistWindows:

Defines equal distant rolling windows.

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

tangencyStrategy:

A pre-defined tangency portfolio strategy.

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

emaSmoother:

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

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

Value

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

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

`emaSmoother`
function returns a numeric vector of smoothed weights.

References

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

`backtest-getMethods` *Portfolio Backtest Extractors*

Description

Extractor functions to get information from objects of class fPFOLIOBACKTEST.

Arguments

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

References

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

Examples

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

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

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

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

---

backtest-performance *Portfolio backtesting net performance*

Description

Displays plot of rebased portfolio performance and summary statistics.

Usage

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

Arguments

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

Value

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

Note

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

References

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

`backtest-plots` *Portfolio backtesting plots*

Description

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

Usage

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

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

Arguments

`object`	a list, returned from running the function `portfolioSmoothing`.
`which`	an integer or string value. If the argument is an integer then it specifies which backtest plot should be displayed. If the argument take the character value `all`, which is the default, then all 6 available backtest plots will be displayed.
`labels`	a logical flag, determining if the graph should be labeled automatically. This is the default case `labels=TRUE`. If set to `FALSE` then the graph will be displayed undecorated and the user can it decorate by himself.

`legend` a logical flag, determining if to the graph a legend should be added. This is the default case `labels=TRUE`. If set to `FALSE` then the graph will be displayed undecorated and the user can it decorate by himself.

`at` if NULL the time-axis ticks will be selected automatically. If `at` is a vector of `timeData` character formatted dates then the axis ticks ar taken from this vector.

`format` if NULL the time-axis ticks are labeled automatically. If `format` is a POSIX format string, tthen the label formats are taken from this string.

`cex, font, family`
font size, font and font family specification for the report.

Details

These backtest plot summarises the results obtained from portfolio backtesting.

The function `backtestAssetsPlot` displays the set of possible assets to construct a portfolio.

The function `backtestWeightsPlot` displays the recommended weights for investment.

The function `backtestRebalancePlot` displays the weight changes over time for individual assets and for the portfolio.

The function `backtestPortfolioPlot` displays the daily, benchmark and portfolio series of a portfolio backtest.

The function `backtestDrawdownPlot` displays the daily drawdowns for the benchmark and the portfolio.

The function `backtestReportPlot` summarises the results from a portfolio backtest.

References

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

---

backtest-portfolio *Portfolio backtesting*

Description

Tests a portfolio by a rolling backtest.

Usage

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

portfolioSmoothing(object, backtest, trace = TRUE)
```

Arguments

formula	a formula describing the benchmark and assets used for backtesting in the form backtest ~ assetA + ... + assetZ. Here, backtest and asset* are column names of the data set.
data	an object of class timeSeries.
spec	an S4 object of class fPFOLIOSPEC as returned by the function portfolioSpec.
constraints	a character string value or vector defining the constraints, for details we refer to portfolioConstraints.
backtest	an S4 object of class fPFOLIOBACKTEST as returned by the function portfolioBacktest.
object	a list as returned by the function portfolioBacktesting.
trace	a logical flag, by default TRUE. Should the backtersting be traced?

References

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

---

backtest-specification

Specification of portfolio backtesting

Description

Specifies how the portfolio backtesting is performed.

Usage

```
portfolioBacktest(
    windows = list(
        windows = "equidistWindows",
        params = list(horizon = "12m")),
    strategy = list(
        strategy = "tangencyStrategy",
        params = list()),
    smoother = list(
        smoother = "emaSmoother",
        params = list(doubleSmoothing = TRUE,
        lambda = "3m", skip = 0,
        initialWeights = NULL)),
    messages = list())
```

Arguments

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

Value

returns an S4 object of class "fPFOLIOBACKTEST".

References

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

---

backtestStats *Rolling portfolio backtesting statistics*

Description

Computes rolling statistics for backtest analysis

Usage

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

rollingSigma(object)
rollingVaR(object)
rollingCVaR(object)
rollingDaR(object)
rollingCDaR(object)
```

Arguments

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

Details

The function rollingSigma calculates the portfolio risk, Sigma, over time.

The function rollingVaR calculates a rolling Value at Risk.

The function rollingCVaR calculates a rolling Conditional Value at Risk.

The function rollingDaR calculates a rolling Drawdowns at Risk.

The function rollingCDaR calculates a rolling Conditional Drawdowns at Risk.

References

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

`data-sets` *Assets Data Sets*

Description

Example data sets for portfolio optimization.

Usage

```
ECON85
ECON85LONG

GCCINDEX
SPISECTOR
SWX
LPP2005
SMALLCAP

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

Value

an object of class `"timeSeries"`.

References

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

fPFOLIOBACKTEST *Portfolio backtesting specifications*

Description

Specifies portfolio backtesting objects.

Usage

```
## S4 method for signature 'fPFOLIOBACKTEST'
show(object)
```

Arguments

object	an S4 object of class fPFOLIOBACKTEST.

Details

Portfolio Backtest Specification:

The S4 class fPFOLIOBACKTEST specifies portfolio backtesting. The slots are:

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

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

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

@messages a list, any messages collected during the backtest

Value

portfolioBacktest returns an S4 object of class "fPFOLIOBACKTEST".

References

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

fPFOLIOCON *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.

fPFOLIODATA *Portfolio Data Handling*

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.

`fPFOLIOSPEC` *Specification of Portfolios*

Description

Specifies portfolios.

Usage

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

Arguments

`object`	an S4 object of class `fPFOLIOSPEC`.

Details

Portfolio Specifcation 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", "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.

---

fPFOLIOVAL *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.

---

fPORTFOLIO *Portfolio Class*

Description

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

Usage

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

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

Arguments

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

Details

Portfolio Class:

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

Value

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

@call	a call, returning the matched function call.
@data	a list with two named elements, series holding the time series data if available, otherwise NA, and statistics, itself a named list with two named elements mu and Sigma holding the vector of means and the matrix of covariances.
@description	a character string, allowing for a brief project description.
@portfolio	a list, containing parameter specifications for the portfolio: weights a numeric vector specifying the portfolio weights, targetReturn a numeric value specifying the target return, targetRisk a numeric value specifying the target risk, targetMean a numeric value specifying the target return determinated with function mean(), targetStdev a numeric value specifying the target risk in standart deviation as risk measure.

`@specification` a list with one named element `spec` which represents an object of class `fPFOLIOSPEC`, including all information about the portfolio specifications, see `PortfolioSpec` for further details.

`@title` a title string.

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

Control Parameters

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

mcSteps [plot] - a number, determining number of Monte Carol 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.

frontier-plot *Efficient Frontier Plot*

Description

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

Usage

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

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

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

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

Arguments

object	an S4 object of class fPORTFOLIO, containing slots call, data, specification, constraints, portfolio, title, description.

frontier	a character string, determining which part of the frontier should be extracted. "both" stands for the full hyperbola, "lower" for all points below the minimum variance return and "upper" for the actual efficient frontier, by default "both".
col	a character string vector, setting the color. For frontierPlot it is a two dimensional a vector; first entry is the upper part of the frontier, second entry the lower, by default "black" and "grey". For the other functions the argument defines the color representation, by default sets the default color is the rainbow palette.
add	a logical value, determining whether the frontier should be added to an existing plot, by default FALSE.
return	a character string denoting which type of return should be plotted. Allowed values for the return are either "mean", or "mu".
risk	a character string denoting which type of risk should be plotted. Allowed values for the risk measure are either "cov", "sigma", "VaR", or "CVaR".
auto	a logical flag denoting if the type of return and risk to be plotted should be selected automatically, by default TRUE.
labels	a logical flag, should the plot be automatically labeled and decorated? By default TRUE.
title	a logical flag, should the plot obtain a default main title and x- and y-labels? By default TRUE.
mcSteps	an integer value, the number of Monte Carlo steps.
xlim, ylim	two numeric vectors with two elelemts , the plot range. If set to NULL the values for the plot ranges are determined automatically.
mText	a character string, representing a marginal text string. If set to NULL the value is taken from the title of the input frontier argument.
twoAssets	a logical flag, if TRUE, then the two assets frontier lines will be drawn.
sharpeRatio	a logical flag, if TRUE, then the Sharpe ratio will be added to the plot.
...	optional arguments to be passed.

Details

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

References

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

`frontier-plotControl` *Frontier Plot Control List*

Description

Allows to modify plot settings for the frontier plot.

Usage

```
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.

frontier-points *Get Frontier Points*

Description

Extracts the risk and return coordinates of the efficient frontier.

Usage

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

Arguments

object	an object of class fPORTFOLIO.
frontier	a character string denoting which part of the efficient portfolio should be extractacted.
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 willbe 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.

mathprog-LP *Mathematical Linear Programming*

Description

Mathematical Linear Programming.

Usage

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

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

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

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

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

Arguments

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

Value

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

References

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

`mathprog-NLP` *Mathematical Non-Linear Programming*

Description

Mathematical Non-Linear Programming.

Usage

```
rsolnpNLP(start, objective,
    lower=0, upper=1, linCons, funCons, control=list())
solnpNLP(start, objective,
    par.lower=NULL, par.upper=NULL,
    eqA=NULL, eqA.bound=NULL,
    ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
    eqFun=list(), eqFun.bound=NULL,
    ineqFun=list(), ineqFun.lower=NULL, ineqFun.upper=NULL,
    control=list())
solnpNLPControl(
    rho=1, outer.iter=400, inner.iter=800, delta=1e-07, tol=1e-08, trace=0)
```

```
rnlminb2NLP(start, objective,
    lower=0, upper=1, linCons, funCons, control=list())
nlminb2NLP(start, objective,
    par.lower=NULL, par.upper=NULL,
    eqA=NULL, eqA.bound=NULL,
    ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
    eqFun=list(), eqFun.bound=NULL,
    ineqFun=list(), ineqFun.lower=NULL, ineqFun.upper=NULL,
    control=list())
nlminb2NLPControl(
    eval.max=500, iter.max=400, trace=0, abs.tol=1e-20, rel.tol=1e-10,
    x.tol=1.5e-08, step.min=2.2e-14, scale=1, R=1, beta.tol=1e-20)
rnlminb2

ramplNLP(start, objective,
    lower=0, upper=1, amplCons, control=list(), ...)
amplNLP()
amplNLPControl(
    solver="minos", project="ampl", trace=FALSE)
```

Arguments

`start`	a numeric vector, the start values.
`objective`	a function object, the function to be optimized.
`lower, upper`	lower and upper bounds.
`linCons`	list of linear constraints: mat, lower, upper.
`funCons`	list of function constraints.
`amplCons`	AMPL constraints.
`control`	control list.
`...`	optional arguments to be passed.
`par.lower, par.upper`	...
`eqA`	...
`eqA.bound`	...
`ineqA`	...
`ineqA.lower,ineqA.upper`	...
`eqFun`	...
`eqFun.bound`	...
`ineqFun`	...
`ineqFun.lower,ineqFun.upper`	...
`rho`	1
`outer.iter`	400

inner.iter	800
delta	1.0e-7
tol	1.0e-8
eval.max	500
iter.max	400
trace	0
abs.tol	1e-20
rel.tol	1e-10
x.tol	1.5e-08
step.min	2.2e-14
scale	1
R	1
beta.tol	1e-20
solver	solver name
project	project name

Value

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

References

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

---

mathprog-QP *Mathematical Linear Programming*

Description

Mathematical Quadratic Programming.

Usage

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

rquadprogQP(objective, lower=0, upper=1, linCons, control=list())
quadprogQP(objective=list(dvec=NULL, Dmat=NULL),
    par.lower=NULL, par.upper=NULL,
    eqA=NULL, eqA.bound=NULL,
    ineqA=NULL, ineqA.lower=NULL, ineqA.upper=NULL,
```

```
    control=list())
quadprogQPControl(solver="quadprog", trace=FALSE)
rquadprog

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

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

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

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

Arguments

objective	...
lower, upper	lower and upper bounds.
linCons	list of linear constraints: mat, lower, upper.
control	control list.
...	optional arguments to be passed.
par.lower, par.upper	...
eqA	...
eqA.bound	...

ineqA	...
ineqA.lower,ineqA.upper	...
x_L,x_U	...
A	...
b_L,b_U	...
solver	...
category	...
project	...
inf	...
trace	...
sigf	...
maxiter	...
margin	...
bound	...
verb	...

Value

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

References

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

`methods-plot` *plot-methods*

Description

Various plot-methods. In particalur, functions `.fportfolio.plot.[i]()` will:

1. plot the efficient frontier,
2. add minimum risk portfolio,
3. add tangency portfolio,
4. add risk/return of single assets,
5. add equal weights portfolio,
6. add two asset frontiers [0-1 PF only],
7. add Monte Carlo portfolios, and/or
8. add Sharpe ratio [MV PF only].

Usage

```
.fportfolio.plot.1(x)
.fportfolio.plot.2(x)
.fportfolio.plot.3(x)
.fportfolio.plot.4(x)
.fportfolio.plot.5(x)
.fportfolio.plot.6(x)
.fportfolio.plot.7(x)
.fportfolio.plot.8(x)
```

Arguments

x	an 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.

---

methods-show *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.

`methods-summary` *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.

`monitor-stability` *Monitoring Stability*

Description

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

Usage

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

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

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

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

parAnalytics()
```

Arguments

index	an object of class 'timeSeries'
method	name of selected analytics
analytics	analytics object
...	optional arguments
spar	0.5
main	""
trace	TRUE
doplot	TRUE
at	pretty()
format	"%m/%y"
lambda	riskmetricsAnalytics
bcp	bcpAnalytics
FUN,strong,k,cs,outbound	pcoutAnalytics
palette,a,b,K	addRainbow

References

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

---

nlminb2 *Constrained nonlinear minimization*

Description

Solve constrained nonlinear minimization problem with nonlinear constraints using a penalty and barrier approach.

Usage

```
nlminb2(start, objective, eqFun = NULL, leqFun = NULL,
    lower = -Inf, upper = Inf, gradient = NULL, hessian = NULL,
    control = list(), env = .GlobalEnv)
```

Arguments

start	a numeric vector, initial values for the parameters to be optimized.
objective	function to be minimized. Must return a scalar value (possibly NA/Inf). The first argument to objective is the vector of parameters to be optimized, whose initial values are supplied through start. Further arguments (fixed during the course of the optimization) to objective may be specified as well. see env.
eqFun	a list of functions describing equal constraints.
leqFun	a list of functions describing less equal constraints.
lower, upper	two vectors of lower and upper bounds, replicated to be as long as start. If unspecified, all parameters are assumed to be unconstrained.
gradient	an optional function that takes the same arguments as objective and evaluates the gradient of objective at its first argument. Must return a vector as long as start.
hessian	an optional function that takes the same arguments as objective and evaluates the hessian of objective at its first argument. Must return a square matrix of order length(start). Only the lower triangle is used.
control	a list of control parameters. See below for details.
env	the environment in which objective, constraint, control functions are evaluated.

Value

A list with following elements:

par	a numeric vector, the best set of parameters found.
objective	a numeric value, the value of objective corresponding to par.
convergence	an integer code, 0 indicates successful convergence.
message	a character string giving any additional information returned by the optimizer, or NULL. For details, see PORT documentation.
iterations	am integer value, the number of iterations performed.
evaluations	an integer value, the number of objective function and gradient function evaluations.

Author(s)

For the R port of nlminb Douglas Bates and Deepayan Sarkar, for the R/Rmetrics port of nlminb2 Diethelm Wuertz, for the PORT library netlib.bell-labs.com.

References

Paul A. Jensen & Jonathan F. Bard, Operations Research Models and Methods, 2001 Appendix A, Algorithms for Constrained Optimization, https://www.me.utexas.edu/~jensen/ORMM/supplements/index.html.

PORT Library, https://netlib.org/port/.

nlminb2Control *Control variables for Rnlminb2*

Description

Collection of Control Variables

Usage

```
nlminb2Control(eval.max = 500, iter.max = 400, trace = 0, abs.tol = 1e-20,
    rel.tol = 1e-10, x.tol = 1.5e-8, step.min = 2.2e-14, scale = 1,
    R = 1.0, beta = 0.01, steps.max = 10, steps.tol = 1e-6)
```

Arguments

eval.max	an integer value. Maximum number of evaluations of the objective function allowed. Defaults to 500.
iter.max	an integer value. Maximum number of iterations allowed. Defaults to 400.
trace	an integer value. The value of the objective function and the parameters is printed every trace'th iteration. Defaults to 0 which indicates no trace information is to be printed.
abs.tol	a numeric value. Absolute tolerance. Defaults to 1e-20.
rel.tol	a numeric value. Relative tolerance. Defaults to 1e-10.
x.tol	a numeric value. X tolerance. Defaults to 1.5e-8.
step.min	a numeric value. Minimum step size. Defaults to 2.2e-14.
scale	See PORT documentation (or leave alone).
R	a numeric value. The multiplier and devisor for the barrier and penalty function terms. Defaults to 1.0
beta	a numeric value. The value by which R is lowered in each iteration step. Defaults to 0.01.
steps.max	an integer value. The maximum number of iteration steps in which the penalty and barrier terms are lowered. Defaults to 10.
steps.tol	a numeric value. The penalty and barrier tolerance. Defaults to 1e-6.

portfolio-constraints *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")

minBuyinConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
maxBuyinConstraints(data, spec=portfolioSpec(), constraints="LongOnly")

nCardConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
minCardConstraints(data, spec=portfolioSpec(), constraints="LongOnly")
maxCardConstraints(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.
...	For internal use only.

Details

How to define constraints?

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

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

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

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

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

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

Lower and Upper Bounds: minW and maxW

Group Constraints: eqsumW, minsumW and maxsumW

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

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

sector constraints `"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 explicitely, 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.

portfolio-covEstimator

Covariance Estimators

Description

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

Usage

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

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

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

covMcdEstimator(x, spec = NULL, ...)
covOGKEstimator(x, spec = NULL, ...)
shrinkEstimator(x, spec = NULL, ...)
nnveEstimator(x, spec = NULL, ...)
```

Arguments

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

Details

The functions are underlying the following algorithms:

covEstimator uses standard covariance estimation,
mveEstimator uses the function "cov.mve" from the MASS package,
mcdEstimator uses the function "cov.mcd" from the MASS package,
lpmEstimator returns lower partial moment estimator,
kendallEstimator returns Kendall's rank estimator,
spearmanEstimator returns Spearman's rankestimator,
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); *ImprovedEestimation 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.

`portfolio-dataSets` *portfolioData2*

Description

portfolioData2.

References

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

`portfolio-efficientPortfolio`
Efficient Portfolios

Description

Returns efficient portfolios.

Usage

```
efficientPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")

maxratioPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
tangencyPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")

minriskPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
minvariancePortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")

maxreturnPortfolio(data, spec = portfolioSpec(), constraints = "LongOnly")
```

Arguments

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

Details

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.

---

portfolio-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 timeSeries. If your timeSerie is not a timeSeries object, consult the generic function as.timeSeries to convert your time series.
spec	an S4 object of class fPFOLIOSPEC as returned by the function portfolioSpec.

Details

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

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

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

Value

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

References

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

`portfolio-getData` *Portfolio Data Extractor Functions*

Description

Extracts information from an object of class fPFOLIODATA.

Usage

```
## S3 method for class 'fPFOLIODATA'
getData(object)
## S3 method for class 'fPFOLIODATA'
getSeries(object)
## S3 method for class 'fPFOLIODATA'
getNAssets(object)
## S3 method for class 'fPFOLIODATA'
getUnits(x)

## S3 method for class 'fPFOLIODATA'
getStatistics(object)
## S3 method for class 'fPFOLIODATA'
getMean(object)
## S3 method for class 'fPFOLIODATA'
getCov(object)
## S3 method for class 'fPFOLIODATA'
getMu(object)
## S3 method for class 'fPFOLIODATA'
getSigma(object)
## S3 method for class 'fPFOLIODATA'
getEstimator(object)

## S3 method for class 'fPFOLIODATA'
getTailRisk(object)
```

Arguments

`object`	an object of class `fPFOLIODATA`.
`x`	an object of class `fPFOLIODATA`.

Details

getData	Extracts data slot,
getSeries	Extracts assets series,
getNAssets	Extracts number of assets,
getUnits	Extracts names of assets,
getStatistics	Extracts statistics slot,
getMean	Extracs mean vector,
getCov	Extracs covariance matrix,
getMu	Extracs mu vector,
getSigma	Extracs Sigma matrix,
getEstimator	Extracs 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.

---

portfolio-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)
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.

References

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

---

portfolio-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'
getUnits(x)
## S3 method for class 'fPORTFOLIO'
getStatistics(object)
## S3 method for class 'fPORTFOLIO'
getMean(object)
## S3 method for class 'fPORTFOLIO'
getCov(object)
## S3 method for class 'fPORTFOLIO'
getMu(object)
## S3 method for class 'fPORTFOLIO'
getSigma(object)
## S3 method for class 'fPORTFOLIO'
getEstimator(object)

## S3 method for class 'fPORTFOLIO'
getSpec(object)
## S3 method for class 'fPORTFOLIO'
getModel(object)
## S3 method for class 'fPORTFOLIO'
getType(object)
## S3 method for class 'fPORTFOLIO'
getOptimize(object)
## S3 method for class 'fPORTFOLIO'
getEstimator(object)
## S3 method for class 'fPORTFOLIO'
getTailRisk(object)
## S3 method for class 'fPORTFOLIO'
getParams(object)
## S3 method for class 'fPORTFOLIO'
```

```
getOptim(object)
## S3 method for class 'fPORTFOLIO'
getSolver(object)
## S3 method for class 'fPORTFOLIO'
getTrace(object)

## S3 method for class 'fPORTFOLIO'
getConstraints(object)

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

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


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

Arguments

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

References

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

`portfolio-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 riks 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.

---

portfolio-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.

---

portfolio-pfolioRisk *portfolioRisk*

Description

Computes covariance and CVaR 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 timeSeries.
weights	a numeric vector of weights.
alpha	a numeric value, the confidence level, by default alpha=0.05, i.e. 5%.

References

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

---

portfolio-portfolioFrontier

Efficient Portfolio Frontier

Description

Compoutes 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 timeSeries. If your timeSerie is not a timeSeries object, consult the generic function as.timeSeries to convert your time series.
description	a character string which allows for a brief description.
include.mvl	a logical flag, should the minimum variance locus be added to the plot?
spec	an S4 object of class fPFOLIOSPEC as returned by the function portfolioSpec.
title	a character string which allows for a project title.

Details

Portfolio Frontier:

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

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

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

Value

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

References

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

`portfolio-portfolioSpec`

Specification of Portfolios

Description

Specifies a portfolio from scratch.

Usage

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

Arguments

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

Details

To optimize a portfolio of assets we first have to specify it. All settings which specify a portfolio of assets are 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 condititional 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 psssible. 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 rquired, and ... additional arguments passed to the users code. Note, myEstimator must a return a named list, with at least the following two entries \$mu and \$Sigma, which represent estimators for the mean and covariance, respectively.

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

\code{"lpmEstimator"},

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

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

Tail Risk List:
The list entry tailRisk from the @model slot is an empty list. It can be used to add tail risk budget constrains to the optimization. In this case a square matrix of the size of the number of assets is expected as list entry, which contains bivariate tail risk measures, i.e. the tail dependence coefficients estaimated 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 ebhanced 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 startet. 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 reqauired 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 availalbe 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 fanction `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 riks 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.

---

portfolio-riskPfolio *Risk and Related Measures for Portfolios*

Description

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

The functions are:

pfolioVaR	computes Value-at-Risk for a portfolio of assets,
pfolioCVaRplus	computes Value-at-Risk+ for a portfolio of assets,
pfolioCVaR	computes Conditional Value-at-Risk for a PF of assets,
lambdaCVaR	computes CVaR's atomic split value lambda,
pfolioCVaRoptim	computes Conditional VaR from mean-CVaR optimization,
pfolioMaxLoss	computes Maximum Loss for a portfolio of assets,
pfolioReturn	computes return values of a portfolio,
pfolioTargetReturn	computes the target return of a portfolio,
pfolioTargetRisk	computes the target risk of a portfolio,
pfolioHist	plots a histogram of the returns of a portfolio.

Usage

```
pfolioVaR(x, weights = NULL, alpha = 0.05)
pfolioCVaRplus(x, weights = NULL, alpha = 0.05)
pfolioCVaR(x, weights = NULL, alpha = 0.05)
lambdaCVaR(n, alpha = 0.05)
pfolioCVaRoptim(x, weights = NULL, alpha = 0.05)

pfolioMaxLoss(x, weights = NULL)
pfolioReturn(x, weights = NULL, geometric = FALSE)
pfolioTargetReturn(x, weights = NULL)
pfolioTargetRisk(x, weights = NULL)
pfolioHist(x, weights = NULL, alpha = 0.05, range = NULL, details = TRUE, ...)
```

Arguments

x	a 'timeSeries' object, data frame or any other rectangular object which can be expressed as a matrix. The first dimension is the number of observations, we call it n, and the second is the number of assets in the data set, we call it dim.
weights	usually a numeric vector which has the length of the number of assets. The weights measures the normalized weights of the individual assets. By default NULL, then an equally weighted set of assets is assumed.
geometric	a logical flag, should geometric returns be used, by default FALSE
alpha	a numeric value, the confidence interval, by default 0.05.
details	a logical value, should details be printed?
n	the number of observation from which the CVaR's atomic split value lambda=1-floor(alpha*n)/(alpha will be evaluated.
range	a numeric vector of two elements limiting the plot range of the histogram. This is quite useful if one likes to compare several plots on the same scale. If range=NULL, the default value, then the range will be selected automatically.
...	optional arguments to be passet to the function hist.

Details

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

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

pfolioCVaRplus or "CVaR+" or the "upper CVaR" are the expected losses strictly exceeding VaR. This is also also called "Mean Excess Loss" and "Expected Shortfall".

pfolioCVaR is a weighted average of VaR and CVaRplus defined as $CVaR = lambda * VaR + (1 - lambda)$ CVaRplus, for $0 <= lambda <= 1$.

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

Value

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

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

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

`lambdaCVaR`
returns CVaR's atomic split value `lambda`, a numeric value.

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

`pfolioReturn`
returns the total portfolio return computed from the set of assets `x`, a numeric vector.

`pfolioTargetReturn`
returns the total return or target return computed from the set of assets `x` and weights `weights`, a numeric value.

`pfolioTargetRisk`
returns the total risk (Sigma) or target risk computed from the set of assets `x` and `weights` via the formual `sqrt(weights %*% cov(x) %*% weights)`, a numeric value.

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

References

Uryasev S. (2000); *Conditional Value-at-Risk (CVaR): Algorithms and Applications*, Risk Management and Financial Engineering Lab, University of Florida

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

portfolio-rollingPortfolios
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

action	a character string naming a user defined function. This function is optionally applied after each rolling step.
by	a character string, by default "1m", which denotes 1 month. The shift by which the portfolio is rolled.
constraints	a character string vector, containing the constraints of the form "minW[asset]=percentage" for box constraints resp. "maxsumW[assets]=percentage" for sector constraints.
data	a list, having a statistics named list, having named entries 'mu' and 'Sigma', containing the information of the statistics.
description	a character string, allowing for a brief project description, by default NULL, i.e. Date and User.
from, to	a vector of S4 timeDate objects which denote the starting and ending dates for the investigation.

period	a character string, by default "12m", which denotes 12 months. The period over which the portfolio is rolled.
spec	an S4 object of class fPFOLIOSPEC.
title	a character string, containing the title for the object, by default NULL.
x	an S4 object of class timeSeries from which the rolling window frames will be created. The length of these frames is given by the argument period and they are shifted by the value specified by the argument by.
...	optional arguments to be passed.

Details

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.

portfolio-setSpec *Settings for Specifications of Portfolios*

Description

Functions to set specifications for a portfolio.

Usage

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

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

setSolver(spec) <- value
setObjective(spec) <- value
setTrace(spec) <- value
```

Arguments

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

Details

setType	Sets type of portfolio optimization,
setOptimize	Sets what to optimize, min risk or max return,
setEstimator	Sets names of mean and covariance estimators,
setParams	Sets optional model parameters,
setWeights	Sets weights vector,
setTargetReturn	Sets target return value,
setTargetRisk	Sets target risk value,

`setTargetAlpha`	Sets CVaR target alpha value,
`setRiskFreeRate`	Sets risk-free rate value,
`setNFrontierPoints`	Sets number of frontier points,
`setStatus`	Sets status value,
`setSolver`	Sets the type of solver to be used,
`setObjective`	Sets objective function name to be used,
`setTrace`	Sets the logical trace flag.

Value

`setType`
`setOptimize`
`setEstimator`
`setParam`

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

`setWeights`
`setTargetReturn`
`setTargetRisk`
`setTargetAlpha`
`setRiskFreeRate`
`setNFrontierPoints`
`setStatus`

Portfolio Settings: just modify the portfolio settings including predefined weights, the target return, the risk free rate, the number of frontier points, and the return and risk range of an existing portfolio structure.

`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.

risk-budgeting *Risk Budgeting*

Description

Functions for risk budgeting.

Usage

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

budgetsSampleCOV(x, weights, mu=NULL, Sigma=NULL)

budgetsNormalVAR(x, weights, alpha=0.05, mu=NULL, Sigma=NULL)
budgetsModifiedVAR(x, weights, alpha=0.05, mu=NULL, Sigma=NULL,
  M3=NULL, M4=NULL)

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

Arguments

x	x
weights	weights
alpha	alpha
mu, Sigma	mean and covariance
M3, M4	M3 and M4

References

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

risk-surfaceRisk *Surface Risk Analytics*

Description

Functions for surface risk analytics.

Usage

```
markowitzHull(data, nFrontierPoints=50)
feasibleGrid(hull, trace=FALSE)
```

Arguments

data	data
hull	hull
nFrontierPoints	nFrontierPoints
trace	trace

References

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

risk-ternaryMap *Creates and Plots a Ternary Map*

Description

Functions for craeting and plotting ternary maps.

Usage

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

riskMap(data, weights)
maxddMap(data, weights)

ternaryWeights(n=21)
ternaryCoord(weights)
ternaryPoints(weights, ...)
```

Arguments

`data`	data
`weights`	weights
`FUN, locator, N, palette, nlevels`	ternaryMap
`n`	n
`...`	optional arguments

References

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

`solve-environment` *Nonlinear Objective Presettings*

Description

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

Usage

```
Data

portfolioObjective(weights)
portfolioReturn(weights)
portfolioRisk(weights)
```

Arguments

`weights`	a vector of portfolio weights

References

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

`solver-ampl` *AMPL Interface*

Description

R/AMPL Interface functions.

Usage

```
amplModelOpen(project)
amplModelAdd(model, project)
amplModelShow(project)

amplDataOpen(project)
amplDataAdd(name, data, type, project)
amplDataAddValue(data, value, project)
amplDataAddVector(data, vector, project)
amplDataAddMatrix(data, matrix, project)
amplDataSemicolon(project)
amplDataShow(project)

amplRunOpen(project)
amplRunAdd(run, project)
amplRunShow(project)

amplOutShow(project)
```

Arguments

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

Value

returns AMPL files.

Author(s)

Diethelm Wuertz.

References

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

`solver-family` *LP, QP, and NLP Programming Solvers*

Description

Rmetrics solver interface.

Usage

```
solveRglpk.CVAR(data, spec, constraints)
solveRglpk.MAD(data, spec, constraints)
solveRampl.CVAR(data, spec, constraints)

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

solveRsocp(data, spec, constraints)


solveRsolnp(data, spec, constraints)
```

Arguments

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

Value

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

References

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

utils-methods *Print Method for Solvers*

Description

S3 print method for mathematical programming solvers.

Usage

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

Arguments

x	x
...	optional arguments

References

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

weights-linePlot *Portfolio Weights Line Plots*

Description

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

Usage

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

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

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

Arguments

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

Details

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

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

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

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

References

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

---

weights-piePlot *Portfolio Pie Plots*

Description

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

Usage

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

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

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

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

Arguments

object	an S4 object of class fPORTFOLIO, as returned by one of the portfolio functions, e.g. efficientPortfolio or portfolioFrontier.
pos	NULL or an integer value. If NULL it is assumend that we consider a single portfolio like for example a tengency portfolio. However, if the object 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 give by the number of frontier points, which can be retrieved by calling getNFrontierPoints.
labels	a logical flag, determining if the graph should be labeled automatically, which is the default case labels=TRUE. If set to FALSE then the graph will be displayed undecorated and the user can it decorate by himself. Evenmore, if labels takes the value of a string vector, then the names of the assets from the porftolio object will be ignored, and the labels will be taken from the specified string vector.
col	a character string vector, defined from a color palette. The default setting uses the "Blues" seqPalette palette.
box	a logical flag, determining whether a boxed frame should be plotted around the pie, by default the value is set to TRUE.
legend	a logical flag, determining if a legend should be added to the plot. The default setting shows the legend.
radius	a numeric value, determining the radius of the pie. The default value is 0.8.
...	arguments to be passed.

Details

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

The function weightsPie displays the weights composition of a portfolio.

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

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

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

References

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

---

weights-Slider *Portfolio Weights Slider*

Description

Interactive portfolio weights plot.

Usage

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

Arguments

control	a list, defining the plotting parameters. The list modifies amongst others the color, e.g. minvariance.col, type of point, e.g. tangency.pch, or the dimension of the point, e.g. cml.cex, see Notes for a complete list of control parameters.
object	an S4 object of class fPORTFOLIO.
...	optional arguments to be passed.

Details

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

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

Value

Creates interactive plots.

Control Parameters

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

singleAsset.cex a number, determining size (percentage) of the singel 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 Carol portfolio symbols, by default 0.01.

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

mcSteps a number, determining number of Monte Carol 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.

`weightsPlot` *Portfolio Weights Bar Plots*

Description

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

Usage

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

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

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

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

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

Arguments

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

Details

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

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

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

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

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

References

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

Index