# Package: Strategy (via r-universe)

August 28, 2024

**Type** Package

**Title** Generic Framework to Analyze Trading Strategies

**Version** 1.0.1

**Date** 2017-08-21

**Author** Julian Busch

**Maintainer** Julian Busch <jb@quants.ch>

**Depends** R (>= 3.2.3)

**Imports** stats, utils, graphics, grDevices, methods, zoo, xts

**Description** Users can build and test customized quantitative trading strategies. Some quantitative trading strategies are already implemented, e.g. various moving-average filters with trend following approaches. The implemented class called ``Strategy'' allows users to access several methods to analyze performance figures, plots and backtest the strategies. Furthermore, custom strategies can be added, a generic template is available. The custom strategies require a certain input and output so they can be called from the Strategy-constructor.

**License** GPL

**LazyData** TRUE

**Suggests** knitr

**VignetteBuilder** knitr

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Date/Publication** 2017-08-24 16:30:36 UTC

**Repository** https://quantsch.r-universe.dev

**RemoteUrl** https://github.com/cran/Strategy

**RemoteRef** HEAD

**RemoteSha** 8e20477270800b831be0ab443ed2d873b44dd8d8

# Contents



---

`assets` *Random walks for 10 assets as example data.*

---

## Description

The dataset contains the price data (not returns!), each starting at a value of 100. The dates are randomly recreated by choosing the latest date as Sys.Date() going backwards on a daily basis per row.

## Usage

```
assets
```

## Format

An xts-object with 1000 rows and 10 variables:

**asset1** Column with price data of a random walk called asset1.

**asset2** Column with price data of a random walk called asset2. ...

---

`backtest` *Backtest Strategy*

---

## Description

Walk forward analysis backtest with the specified parameters on an object of class `Strategy`. The backtest calibrates the parameters according to the specification given by the user (in-sample) and returns the trading signals for the following period (out-of-sample). This is iteratively repeated on a shifting time window. Computer performance is critical with this function.

## Usage

```
backtest(object, horizon = "6m", data.width = "24m", keep.history = F,
  optim.param = NULL, optim.param.min = 1, optim.param.max = 10,
  optim.param.scale = 0.1, from = NULL, until = NULL, which = NULL,
  rf = 0, printSteps = F)

## S4 method for signature 'Strategy'
backtest(object, horizon = "6m", data.width = "24m",
  keep.history = F, optim.param = NULL, optim.param.min = 1,
  optim.param.max = 10, optim.param.scale = 0.1, from = NULL,
  until = NULL, which = NULL, rf = 0, printSteps = F)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `horizon` | The out-of-sample period length. |
| `data.width` | The in-sample period length used for calibration. |
| `keep.history` | If set to `TRUE`, the starting point of in-sample data is kept fixed, so the period extends each iteration. |
| `optim.param` | A character vector providing the names of the parameters to be calibrated. Parameters that are not provided will be kept fix. |
| `optim.param.min` | A numeric vector providing the minimum values of the parameters that are calibrated. |
| `optim.param.max` | A numeric vector providing the maximum values of the parameters that are calibrated. |

optim.param.scale
: A numeric vector providing the scaling of the parameters that are calibrated. It is advisable to set scaling of the parameters to the smallest unit that makes sense.

from
: The date in character format "yyyy-MM-dd" or as date-object from which assets shall be considered. If NULL, no restriction is made.

until
: The date in character format "yyyy-MM-dd" or as date-object until which assets shall be considered. If NULL, no restriction is made.

which
: Names or number of assets that should be included in backtest

rf
: Risk free rate in decimal, e.g. rf=0.01 equals 1 percent.

printSteps
: This is a feature used mainly for debugging the constructor function in order to localize where unspecified errors occur. If set to true, the different steps run within the constructor is printed to the console.

## Examples

```
##Not run:
# MA(200)-Strategy
params <- list(k=20)
# reduce dataset due to computation time
assets_r <- assets[tail(zoo::index(assets),100)]
myStrat.MA <- Strategy(assets=assets_r, strat="MA", strat.params=params)

# Perform backtest on MA(20)-Strategy with
# out-of-sample periods of 2 months
# and in-sample-calibration of 2 months
# This example requires a lot of computation time,
# so this is only performed for 1 asset and high scaling.
backtest(myStrat.MA, horizon="2m", data.width="2m"
         , optim.param="k", optim.param.min=5, optim.param.max=10
         , optim.param.scale=5, printSteps = TRUE, which=1)
##End(Not run)
```

---

# compare *Compare performance of* Strategy*-objects.*

---

## Description

Compare the portfolio performance indicators of an arbitrary number of objects of class Strategy.

## Usage

```
compare(..., from=NULL, until=NULL, which=NULL
       , scaling.periods=NULL, include.costs=TRUE
       , use.backtest=FALSE, include.params=FALSE)

## S4 method for signature 'Strategy'
compare(..., from = NULL, until = NULL, which = NULL,
  scaling.periods = NULL, include.costs = TRUE, use.backtest = FALSE,
  include.params = FALSE)
```

## Arguments

| | |
|---|---|
| ... | Objects of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which performance shall be considered. If NULL, no restriction is made. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which performance shall be considered. If NULL, no restriction is made. |
| which | Names or number of assets that should be included in calculation. |
| scaling.periods | Vector with annualization factors for calculation. Default is 252, 52, 12, 4, 1 for daily, weekly, monthly, quarterly and yearly data respectively. |
| include.costs | If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is obsolete if no costs are given. |
| use.backtest | If TRUE, the performance of the backtesting output is considered for performance indicator calculation. If FALSE, the performance of the initial strategy execution are used. |
| include.params | If TRUE the parameters of the strategies are included in their names. E.g. MA(k=200) instead of MA as strategy name for moving average. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# EWMA(0.05)-Strategy
params <- list(lambda=0.05)
myStrat.EWMA <- Strategy(assets=assets, strat="EWMA", strat.params=params)

# Compare annualized performance of MA(200) and EWMA(0.05)
# compare(myStrat.MA, myStrat.EWMA, use.backtest=TRUE, scaling.periods=252)

##End(Not run)
```

---

# ES *Expected Shortfall*

---

## Description

Expected Shortfall of the assets or portfolio of an object of class Strategy.

## Usage

```
ES(object, alpha=0.05, V=1
       , type="normal.distribution", method="full"
       , of="portfolio", from=NULL, until=NULL, which=NULL
       , scaling.periods=NULL, include.weights=TRUE
       , include.costs=TRUE, use.backtest=FALSE)

## S4 method for signature 'Strategy'
ES(object, alpha = 0.05, V = 1,
  type = c("normal.distribution", "historical"), method = c("full",
  "linear"), of = c("portfolio", "assets"), from = NULL, until = NULL,
  which = NULL, scaling.periods = NULL, include.weights = TRUE,
  include.costs = TRUE, use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `alpha` | The significance level $\alpha$ that is used for probability of cumulative loss at level $1 - \alpha$. |
| `V` | Volume that is invested. The linear factor for the ES calculation. Either a single value for portfolio or a vector for each asset. |
| `type` | Type of ES calculation. Use `normal.distribution` for the normal distribution, `historical` for the empirical distribution. |
| `method` | Method of loss calculation. Use `linear` for approximation with log returns or `full` for calculation with arithmetic returns. |
| `of` | ES to be calculated for assets separately or the portfolio. |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which losses shall be considered. If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which losses shall be considered. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in calculation. |
| `scaling.periods` | Vector with annualization factors for calculation. Default is 252, 52, 12, 4, 1 for daily, weekly, monthly, quarterly and yearly data respectively. |
| `include.weights` | Only relevant if `of="assets"`: If `FALSE`, weights are all set to `1`. This might be necessary if only single stock performance without weighting shall be considered. |
| `include.costs` | If `FALSE`, the fixed and relative trading costs are NOT considered for ES calculation. Default value is `TRUE`. As default values for costs are `0`, this argument is obsolete if no costs are given. |
| `use.backtest` | If `TRUE`, the performance of the backtesting output is considered for VaR calculation. If `FALSE`, the performance of the initial strategy execution are used. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get ES of MA(200)-Strategy portfolio
ES(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Get backtest ES of MA(200)-Strategy (backtest would need to be executed first!)
# ES(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE)

##End(Not run)
```

---

# getBacktestSetup *Get backtest parameter values from* Strategy*-object*

---

## Description

Gets the backtest parameter values of an object of class Strategy that were used for backtesting the strategy. This includes the information about the parameters,

## Usage

```
getBacktestSetup(object)

## S4 method for signature 'Strategy'
getBacktestSetup(object)
```

## Arguments

| | |
|---|---|
| object | An object of class Strategy. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get backtest setup from MA(200)-Strategy
getBacktestSetup(myStrat.MA)

##End(Not run)
```

## getCosts *Get strategy function from* Strategy*-object*

### Description

Returns the fixed and relative trading costs of an object of class Strategy..

### Usage

```
getCosts(object)

## S4 method for signature 'Strategy'
getCosts(object)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get strategy function from MA(200)-Strategy
MA.costs <- getCosts(myStrat.MA)
# return fix costs
MA.costs$fix
# return relative costs
MA.costs$relative

##End(Not run)
```

## getFilters *Get strategy values from* Strategy*-object*

### Description

Gets the strategy values of an object of class Strategy that were output from strategy calculation.

### Usage

```
getFilters(object, which = NULL)

## S4 method for signature 'Strategy'
getFilters(object, which = NULL)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| which | Which filters shall be returned. Either list number or names to be passed. |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get strategy values from MA(200)-Strategy
getFilters(myStrat.MA) # all strategy values returned

##End(Not run)
```

---

## getIndicators *Get indicators from* Strategy*-object*

---

### Description

Gets the indicators data of an object of class Strategy that was used within strategy calculation.

### Usage

```
getIndicators(object, from = NULL, until = NULL, which = NULL)

## S4 method for signature 'Strategy'
getIndicators(object, from = NULL, until = NULL,
  which = NULL)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which indicators shall be returned. If NULL, no restriction is made. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which indicators shall be returned. If NULL, no restriction is made. |
| which | Names or list-number of indicators that should be included. If NULL, all indicators are returned. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
randreturns <- xts::xts(rnorm(nrow(assets)), order.by=
seq(from=Sys.Date()-nrow(assets)+1, to=Sys.Date(), by="d"))
indicators <- list(returns=randreturns) # example: random returns
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params, indicators=indicators)

# Get indicator data from MA(200)-Strategy
getIndicators(myStrat.MA, from="2015-01-01", until="2015-12-31")

##End(Not run)
```

---

`getParameters` *Get strategy function parameters from* `Strategy`*-object*

---

## Description

Gets the strategy function parameters of an object of class `Strategy` that were used for strategy calculation.

## Usage

```
getParameters(object, use.backtest = FALSE)

## S4 method for signature 'Strategy'
getParameters(object, use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `use.backtest` | If set to `TRUE`, the calibrated parameters of the backtest are returned. Requires `backtest` to be executed first. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get parameters from MA(200)-Strategy
getParameters(myStrat.MA)

##End(Not run)
```

## getPrices *Get price data from* Strategy*-object*

### Description

Gets the price data of an object of class Strategy that was used within strategy calculation.

### Usage

```
getPrices(object, from = NULL, until = NULL, which = NULL)

## S4 method for signature 'Strategy'
getPrices(object, from = NULL, until = NULL,
  which = NULL)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which prices shall be returned. If NULL, no restriction is made. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which prices shall be returned. If NULL, no restriction is made. |
| which | Names or column-number of assets that should be included. If NULL, all prices are returned. |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get price data from MA(200)-Strategy
getPrices(myStrat.MA, from="2015-01-01", until="2015-12-31")

##End(Not run)
```

## getSignals *Get trading signals from* Strategy*-object*

### Description

Gets the trading signals of an object of class Strategy that were output from strategy calculation.

### Usage

```
getSignals(object, from = NULL, until = NULL, which = NULL,
  use.backtest = FALSE)

## S4 method for signature 'Strategy'
getSignals(object, from = NULL, until = NULL,
  which = NULL, use.backtest = FALSE)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which signals shall be returned. If NULL, no restriction is made. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which signals shall be returned. If NULL, no restriction is made. |
| which | Names or column-number of assets that should be returned. If NULL, all signals are returned. |
| use.backtest | If set to TRUE, the signals of the backtest are returned. Requires backtest to be executed first. |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get signals from MA(200)-Strategy
# all signals returned
getSignals(myStrat.MA)
# backtest signals for first two assets returned
# getSignals(myStrat.MA, which=c(1,2), use.backtest=TRUE)

##End(Not run)
```

---

## getStratFUN *Get strategy function from* Strategy*-object*

---

### Description

Gets the strategy function of an object of class Strategy that was used for strategy calculation.

### Usage

```
getStratFUN(object)

## S4 method for signature 'Strategy'
getStratFUN(object)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get strategy function from MA(200)-Strategy
MA.FUN <- getStratFUN(myStrat.MA)

##End(Not run)
```

---

## getStratName *Get strategy function name from* Strategy*-object*

---

### Description

Gets the strategy function name of an object of class Strategy that was used for strategy calculation. This function is for aesthetic purposes only and does not have any numerical relevance.

### Usage

```
getStratName(object, include.params = FALSE)

## S4 method for signature 'Strategy'
getStratName(object, include.params = FALSE)
```

### Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| include.params | If set to TRUE, the parameters used for strategy evaluation are included. |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get strategy function name from MA(200)-Strategy
getStratName(myStrat.MA) # returns "MA"
getStratName(myStrat.MA, include.params=TRUE) # returns "MA(200)"

##End(Not run)
```

getTrades *Get trades according to the signals from the* Strategy*-object*

## Description

Gets the trades of an object of class Strategy that were performed within strategy calculation.

## Usage

```
getTrades(object, from = NULL, until = NULL, which = NULL,
  of = "signals", use.backtest = FALSE)

## S4 method for signature 'Strategy'
getTrades(object, from = NULL, until = NULL,
  which = NULL, of = c("signals", "weights"), use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which trades shall be returned. If NULL, no restriction is made. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which trades shall be returned. If NULL, no restriction is made. |
| which | Names or column-number of assets that should be included. If NULL, trades for all assets are returned. |
| of | Trades to be calculated on basis of trading signals or weights of portfolio. |
| use.backtest | If set to TRUE, the trades of the backtest are returned. Requires backtest to be executed first. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get price data from MA(200)-Strategy
getTrades(myStrat.MA, from="2015-01-01", until="2015-12-31")

##End(Not run)
```

getWeights *Get weights from* Strategy*-object*

## Description

Gets the weights data of an object of class Strategy that was used within strategy calculation.

## Usage

```
getWeights(object, from = NULL, until = NULL, which = NULL,
  use.backtest = FALSE)

## S4 method for signature 'Strategy'
getWeights(object, from = NULL, until = NULL,
  which = NULL, use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which weights shall be returned If NULL, no restriction is made. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which weights shall be returned. If NULL, no restriction is made. |
| which | Names or column-number of assets that should be included. If NULL, all weights are returned. |
| use.backtest | If set to TRUE, the weights of the backtest are returned. Requires backtest to be executed first. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get weights data from MA(200)-Strategy
getWeights(myStrat.MA, from="2015-01-01", until="2015-12-31")

##End(Not run)
```

`hitratio` *Strategy Hit Ratio*

## Description

Gets the hitratio of the signals of an object of class `Strategy`.

## Usage

```
hitratio(object, of="portfolio"
       , from=NULL, until=NULL, which=NULL
       , type="per.signal", include.costs=TRUE
       , use.backtest=FALSE)

## S4 method for signature 'Strategy'
hitratio(object, of = c("portfolio", "assets"),
  from = NULL, until = NULL, which = NULL, type = c("per.signal",
  "per.trade"), include.costs = TRUE, use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `of` | Hit Ratio to be calculated for assets separately or the portfolio (weighted hit ratios according to average asset weights). |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which returns shall be considered. If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which returns shall be considered. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in calculation. |
| `type` | If the hitratio shall be calculated per trade with `per.trade` or per signal `per.signal`. |
| `include.costs` | If `FALSE`, the fixed and relative trading costs are NOT considered for performance calculation. Default value is `TRUE`. As default values for costs are `0`, this argument is obsolete if no costs are given. |
| `use.backtest` | If set to `TRUE`, the signals from the backtesting output are considered for maximum drawdown calculation. If `FALSE`, the signals from the initial strategy execution are used. |

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)
```

```
# Get hit ratio of MA(200)-Strategy portfolio
hitratio(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Get hit ratio of MA(200)-Strategy (daily data = 252 trading days)
# hitratio(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE)

## End(Not run)
```

---

`loss` *Get the losses of assets or portfolio over time.*

---

## Description

Losses over time of an assets or portfolio of an object of class `Strategy`.

## Usage

```
loss(object, V=100, method="full", of="portfolio"
       , from=NULL, until=NULL, which=NULL
       , include.weights=TRUE, include.costs=TRUE
       , use.backtest=FALSE)

## S4 method for signature 'Strategy'
loss(object, V = 100, method = c("full", "linear"),
  of = c("portfolio", "assets"), from = NULL, until = NULL,
  which = NULL, include.weights = TRUE, include.costs = TRUE,
  use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `V` | Volume that is invested. The linear factor for the VaR calculation. Either a single value for portfolio or a vector for each asset. |
| `method` | Method of loss calculation. Use `linear` for approximation with log returns or `full` for calculation with arithmetic returns. |
| `of` | Losses to be calculated for assets separately or the portfolio. |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which losses shall be considered. If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which losses shall be considered. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in calculation. |
| `include.weights` | Only relevant if `of="assets"`: If `FALSE`, weights are all set to `1`. This might be necessary if only single stock performance without weighting shall be considered. |

include.costs If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is obsolete if no costs are given.

use.backtest If TRUE, the performance of the backtesting output is considered for loss calculation. If FALSE, the performance of the initial strategy execution are used.

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get VaR of MA(200)-Strategy portfolio
myStrat.MA.losses <- loss(myStrat.MA, from="2015-01-01", until="2015-12-31")

## End(Not run)
```

---

# MDD *Strategy Performance Maximum Drawdown*

---

## Description

Gets the maximum drawdown of the performance of an object of class Strategy.

## Usage

```
MDD(object, of="portfolio"
       , from=NULL, until=NULL, which=NULL
       , type="relative", include.costs=TRUE
       , use.backtest=FALSE)

## S4 method for signature 'Strategy'
MDD(object, of = c("portfolio", "assets"), from = NULL,
  until = NULL, which = NULL, type = c("absolute", "relative"),
  include.costs = TRUE, use.backtest = FALSE)
```

## Arguments

object An object of class Strategy.

of Maximum Drawdown to be calculated for assets separately or the portfolio.

from The date in character format "yyyy-MM-dd" or as date-object from which performance shall be considered. If NULL, no restriction is made.

until The date in character format "yyyy-MM-dd" or as date-object until which performance shall be considered. If NULL, no restriction is made.

which Names or number of assets that should be included in calculation.

| | |
|---|---|
| type | If the absolute or relative drawdown of the performance shall be returned. |
| include.costs | If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is obsolete if no costs are given. |
| use.backtest | If set to TRUE, the signals from the backtesting output are considered for maximum drawdown calculation. If FALSE, the signals from the initial strategy execution are used. |

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get MDD of MA(200)-Strategy portfolio
MDD(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Get MDD of MA(200)-Strategy (daily data = 252 trading days)
# MDD(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE)

## End(Not run)
```

---

# newStrategyFunction *Create Own Strategy*

---

## Description

Creates a strategy function template file. This file can be used as template for the development of customized strategies.

## Usage

```
newStrategyFunction(name = NULL, file.path = getwd(), overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| name | String as name of the new function (without spaces). |
| file.path | Valid file path of existing directory where the new function shall be stored in format file.path/name.R. |
| overwrite | If the strategy file already exists, it will be overwritten if value is TRUE. |

## Examples

```
##Not run:

# Creates a file myNewStrat.R at the specific file path
newStrategyFunction(name="myNewStrat", file.path=getwd(), overwrite=T)

##End(Not run)
```

---

`performance` *Get Strategy Performance*

---

## Description

Gets the performance of an object of class `Strategy`.

## Usage

```
performance(object, of = "portfolio", type = "performance", from = NULL,
  until = NULL, which = NULL, use.backtest = FALSE,
  include.costs = TRUE)

## S4 method for signature 'Strategy'
performance(object, of = c("portfolio", "assets"),
  type = c("performance", "logReturns", "returns"), from = NULL,
  until = NULL, which = NULL, use.backtest = FALSE,
  include.costs = TRUE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `of` | Performance to be extracted from assets separately or the portfolio performance. |
| `type` | Which type of performance shall be returned. `performance` is the cumulative performance starting at `1`, `logReturns` to get logarithmic returns or `returns` for arithmetic returns. |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which performance shall be returned If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which performance shall be returned. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in performance. If a portfolio performance from only a subset of the assets is calculated, the weights are scaled accordingly. |
| `use.backtest` | If `TRUE`, the signals from the backtesting output are considered for performance calculation. If `FALSE`, the signals from the initial strategy execution are used. |
| `include.costs` | If `FALSE`, the fixed and relative trading costs are NOT considered for performance calculation. Default value is `TRUE`. As default values for costs are `0`, this argument is obsolete if no costs are given. |

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get performance of MA(200)-Strategy
performance(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Get backtest performance of MA(200)-Strategy
# performance(myStrat.MA, from="2015-01-01", until="2015-12-31"
# , use.backtest=TRUE, type="logReturns")

## End(Not run)
```

---

# performanceIndicators *Strategy Performance Indicators*

---

## Description

Get a list of the performance indicators of an object of class `Strategy`.

## Usage

```
performanceIndicators(object, of="portfolio"
       , from=NULL, until=NULL, which=NULL, alpha=0.05
       , scaling.periods=NULL, include.weights=TRUE
       , include.costs=TRUE, use.backtest=FALSE)

## S4 method for signature 'Strategy'
performanceIndicators(object, of = c("portfolio",
  "assets"), from = NULL, until = NULL, which = NULL, alpha = 0.05,
  scaling.periods = NULL, include.weights = TRUE, include.costs = TRUE,
  use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `of` | Indicators to be calculated for assets separately or the portfolio. |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which performance shall be considered. If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which performance shall be considered. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in calculation. |
| `alpha` | The significance level $\alpha$ that is used for propability of cumulative loss at level $1 - \alpha$. |

scaling.periods
: Vector with annualization factors for calculation. Default is 252, 52, 12, 4, 1 for daily, weekly, monthly, quarterly and yearly data respectively.

include.weights
: Only relevant if of="assets": If FALSE, weights are all set to 1. This might be necessary if only single stock performance without weighting shall be considered.

include.costs
: If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is obsolete if no costs are given.

use.backtest
: If set to TRUE, the signals from the backtesting output are considered for maximum drawdown calculation. If FALSE, the signals from the initial strategy execution are used.

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get performance indicators of MA(200)-Strategy assets
performanceIndicators(myStrat.MA, from="2015-01-01", until="2015-12-31")

## End(Not run)
```

---

# plot *Plot of a* Strategy*-object*

---

## Description

Calls a generic plot function that can plot the data of any Strategy-object. If a plotFUN-function is given within the object, this user-defined function will be used. The generic function plots 3 parts:

- Price area Plots the asset price data and filters.
- Indicator area Plots the indicators and trading signals.
- Performance area Plots performance of the strategy.

## Usage

```
## S3 method for class 'Strategy'
plot(x, y, from=NULL, until=NULL
       , which.assets=NULL, which.filters=NULL, which.indicators=NULL
       , main=NULL, show.signals=TRUE, include.costs=TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class Strategy. |
| y | Standard plot argument that is not relevant for Strategy objects! |
| from | From date that chart is to be plotted. |
| until | Until date that chart is to be plotted. |
| which.assets | Which assets shall be plotted (each one will result in single plot) |
| which.filters | Which filters shall be added to price plot. Default value NULL will return all filters from the strategy. |
| which.indicators | Which indicators shall be added to indicator plot. Default value NULL will return all filters from the strategy. If "none", no indicator is plotted and indicator area is not shown. |
| main | The main title of the plot. |
| show.signals | If TRUE, the trading signals are shown within the indicators area of the plot. Default value is TRUE. |
| include.costs | If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is redundant if no costs are given. |
| ... | Further arguments passed to custom plotFUN (if available) of the object (x). |

### Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Plot first asset of MA(200)-Strategy
plot(myStrat.MA, from="2015-01-01", until="2015-12-31", which.assets=1)

##End(Not run)
```

---

## plotDrawdowns *Plot Strategy Drawdowns*

---

### Description

Plots drawdowns of the performance of an object of class Strategy.

## Usage

```
plotDrawdowns(object, from = NULL, until = NULL, which = NULL,
  of = "portfolio", type = "relative", include.costs = TRUE,
  use.backtest = FALSE, returnValues = FALSE, ...)

## S4 method for signature 'Strategy'
plotDrawdowns(object, from = NULL, until = NULL,
  which = NULL, of = c("portfolio", "assets"), type = c("relative",
  "absolute"), include.costs = TRUE, use.backtest = FALSE,
  returnValues = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which drawdowns shall be plotted. If NULL, the start date of the performances is used. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which drawdowns shall be plotted. If NULL, the end date of the performances is used. |
| which | Names or number of assets that should be included in performance. If a portfolio performance from only a subset of the assets is calculated, the weights are scaled accordingly. |
| of | Performance to be extracted from assets separately or the portfolio performance. |
| type | If the absolute or relative drawdown of the performance shall be returned. |
| include.costs | If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is redundant if no costs are given. |
| use.backtest | If TRUE, the signals from the backtesting output are considered for drawdowns calculation. If FALSE, the signals from the normal strategy execution with the initial parameters are used. |
| returnValues | If TRUE, the drawdown values are returned. |
| ... | Further arguments that can be passed to the underlying plot()-function. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Plot MA(200)-Strategy drawdowns
plotDrawdowns(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Plot backtested MA(200)-Strategy drawdowns
# plotDrawdowns(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE)

##End(Not run)
```

plotPerformance *Plot Strategy Performance*

## Description

Plots performance of an object of class Strategy.

## Usage

```
plotPerformance(object, which = NULL, of = "portfolio", from = NULL,
  until = NULL, use.backtest = FALSE, include.costs = TRUE,
  plot.params = TRUE, plot.params.names = NULL, plot.params.first = TRUE,
  ...)

## S4 method for signature 'Strategy'
plotPerformance(object, which = NULL,
  of = c("portfolio", "assets"), from = NULL, until = NULL,
  use.backtest = FALSE, include.costs = TRUE, plot.params = TRUE,
  plot.params.names = NULL, plot.params.first = TRUE, ...)
```

## Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| which | Names or number of assets that should be included in performance. If a portfolio performance from only a subset of the assets is calculated, the weights are scaled accordingly. |
| of | Performance to be extracted from assets separately or the portfolio performance. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which performance shall be plotted. If NULL, the start date of the performances is used. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which performance shall be plotted. If NULL, the end date of the performances is used. |
| use.backtest | If TRUE, the signals from the backtesting output are considered for performance calculation. If FALSE, the signals from the normal strategy execution with the intial parameters are used. |
| include.costs | If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is redundant if no costs are given. |
| plot.params | If set to TRUE, the parameters used for the performance periods are plotted into the chart. Requires that use.backtest is set to TRUE. |
| plot.params.names | New parameter names to be shown can be supplied. Requires that use.backtest is set to TRUE to take effect. |
| plot.params.first | If TRUE, the parameter for the first period is plotted. Otherwise, the parameters are plot at the point on the x-axis, from which they are valid. Requires that use.backtest is set to TRUE to take effect. |

| | |
|---|---|
| ... | Further arguments that can be passed to the underlying plot()-function. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Plot MA(200)-Strategy
plotPerformance(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Plot backtested MA(200)-Strategy
# plotPerformance(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE)

##End(Not run)
```

---

# plotWeights *Plot Strategy Weights*

---

## Description

Plots the weights of the portfolio of an object of class Strategy.

## Usage

```
plotWeights(object, from = NULL, until = NULL, ...)

## S4 method for signature 'Strategy'
plotWeights(object, from = NULL, until = NULL, ...)
```

## Arguments

| | |
|---|---|
| object | An object of class Strategy. |
| from | The date in character format "yyyy-MM-dd" or as date-object from which weights shall be plotted. If NULL, the start date of the weights is used. |
| until | The date in character format "yyyy-MM-dd" or as date-object until which weights shall be plotted. If NULL, the end date of the performances is used. |
| ... | Currently not active. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)
```

```
# Plot MA(200)-Strategy weights
plotWeights(myStrat.MA)

##End(Not run)
```

---

sharpe *Get Sharpe Ratio of Performance*

---

## Description

Get the sharpe ratio of the performance of an object of class `Strategy`.

## Usage

```
sharpe(object, rf=0, of="portfolio"
     , from=NULL, until=NULL, which=NULL
     , scaling.periods=NULL, include.costs=TRUE
     , use.backtest=FALSE)

## S4 method for signature 'Strategy'
sharpe(object, rf = 0, of = c("portfolio", "assets"),
  from = NULL, until = NULL, which = NULL, scaling.periods = NULL,
  include.costs = TRUE, use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `rf` | Risk free rate in decimal, e.g. `rf=0.01` equals `1 percent`. |
| `of` | Sharpe ratio to be calculated for assets separately or the portfolio sharpe. |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which performance shall be considered. If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which performance shall be considered. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in calculation. |
| `scaling.periods` | Vector with annualization factors for sharpe ratio calculation. Default is 252, 52, 12, 4, 1 for daily, weekly, monthly, quarterly and yearly data respectively. |
| `include.costs` | If `FALSE`, the fixed and relative trading costs are NOT considered for performance calculation. Default value is `TRUE`. As default values for costs are `0`, this argument is obsolete if no costs are given. |
| `use.backtest` | If `TRUE`, the performance of the backtesting output is considered for sharpe ratio calculation. If `FALSE`, the performance of the initial strategy execution are used. |

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get sharpe of MA(200)-Strategy portfolio
sharpe(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Get backtest annualized sharpe of MA(200)-Strategy (daily data = 252 trading days)
# sharpe(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE, scaling.periods=252)

## End(Not run)
```

---

`Strategy` *Create Strategy Object*

---

## Description

Creates an object of class `Strategy` with the given portfolio data and strategy-function.

## Usage

```
Strategy(assets, strat = "buyhold"
    , assetValueType = c("price", "logReturn"), weights = NULL, indicators = list()
      , strat.params = list(), volume = 1000000
      , costs.fix = 0, costs.rel = 0
      , printSteps = FALSE)
```

## Arguments

| | |
|---|---|
| `assets` | Time series of class `xts` of asset values in either price or log return form on which the strategy function shall be applied. This is the portfolio of assets. |
| `strat` | The name of the strategy that should be applied. This can be either a predefined strategy like MA or EWMA or a self-written function in which case the full path to the function file to be called must be supplied. |
| `assetValueType` | Assets can be passed as prices or log returns. In order to identify the asset value types, either one of the types has to be selected. |
| `weights` | The portfolio weights for the given assets as time series (dynamic) or numerical (constant) weights. |
| `indicators` | A list of indicators that might be used within customized strategies. It is recommended to pass a named list. |
| `strat.params` | The list of parameters and their values required by the strategy function selected with parameter strat. |
| `volume` | Portfolio volume for trading. Default value is 1 Million. |

| | |
|---|---|
| costs.fix | The fix trading costs per trade. |
| costs.rel | The trading costs, relative to the volume. I.e. a value of 10E-4 reflects the costs of 10 basis points of the traded volume. |
| printSteps | This is a feature used mainly for debugging the constructor function in order to localize where unspecified errors occur. If set to true, the different steps run within the constructor is printed to the console. |

## Examples

```
##Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Own MA-strategy-function
# myStrat.MA <- Strategy(assets=assets, strat="C:/MA_function.R")

##End (Not run)
```

---

# Strategy-class Strategy-*Class*

---

## Description

An S4 class to store quantitative strategies and compute various performance figures.

## Slots

prices Price data of the assets. If return data was given within the constructor, starting at 100.

weights Time series of class xts indicating row wise weights of the assets.

indicators List of indicators of class xts.

strat Name of the strategy function to be called. Could be a full file path to a custom strategy.

strat.params List of parameters as input for the strategy function. List entry names should match parameter names.

stratFUN Contains the custom strategy function or NULL.

plotFUN Contains the custom strategy function or NULL.

filters List with filtered price data (e.g. MA(200)-data).

signals Time series with trading signals of class xts.

backtest.signals Time series with trading signals of the backtest of class xts.

backtest.parameters List of parameters of the backtest.

backtest.setup Matrix showing the backtest preferences.

volume Numeric vector indicating the initial investment volume per asset.

costs.fix Numeric vector indicating the fixed costs per trade per asset.

costs.rel Numeric vector indicating the relative costs per trade per asset.

VaR *Value at Risk*

## Description

Value at Risk of the assets or portfolio of an object of class `Strategy`.

## Usage

```
VaR(object, alpha=0.05, V=1, type="normal.distribution"
       , method="full", of="portfolio"
       , from=NULL, until=NULL, which=NULL
       , scaling.periods=NULL, include.weights=TRUE
       , include.costs=TRUE, use.backtest=FALSE)

## S4 method for signature 'Strategy'
VaR(object, alpha = 0.05, V = 1,
  type = c("normal.distribution", "historical"), method = c("full",
  "linear"), of = c("portfolio", "assets"), from = NULL, until = NULL,
  which = NULL, scaling.periods = NULL, include.weights = TRUE,
  include.costs = TRUE, use.backtest = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `Strategy`. |
| `alpha` | The significance level $\alpha$ that is used for propability of cumulative loss at level $1 - \alpha$. |
| `V` | Volume that is invested. The linear factor for the VaR calculation. Either a single value for portfolio or a vector for each asset. |
| `type` | Type of VaR calculation. Use `normal.distribution` for the normal distribution, `historical` for the empirical distribution. Default value is `historical`. |
| `method` | Method of loss calculation. Use `linear` for approximation with log returns or `full` for calculation with arithmetic returns. Default value is `full`. |
| `of` | VaR to be calculated for assets separately or the portfolio. |
| `from` | The date in character format `"yyyy-MM-dd"` or as date-object from which losses shall be considered. If `NULL`, no restriction is made. |
| `until` | The date in character format `"yyyy-MM-dd"` or as date-object until which losses shall be considered. If `NULL`, no restriction is made. |
| `which` | Names or number of assets that should be included in calculation. |
| `scaling.periods` | Vector with annualization factors for calculation. Default is 252, 52, 12, 4, 1 for daily, weekly, monthly, quarterly and yearly data respectively. |
| `include.weights` | Only relevant if `of="assets"`: If `FALSE`, weights are all set to `1`. This might be necessary if only single stock performance without weighting shall be considered. |

include.costs If FALSE, the fixed and relative trading costs are NOT considered for performance calculation. Default value is TRUE. As default values for costs are 0, this argument is obsolete if no costs are given.

use.backtest If TRUE, the performance of the backtesting output is considered for VaR calculation. If FALSE, the performance of the initial strategy execution are used.

## Examples

```
## Not run:

# MA(200)-Strategy
params <- list(k=200)
myStrat.MA <- Strategy(assets=assets, strat="MA", strat.params=params)

# Get VaR of MA(200)-Strategy portfolio
VaR(myStrat.MA, from="2015-01-01", until="2015-12-31")

# Get backtest VaR of MA(200)-Strategy
# VaR(myStrat.MA, from="2015-01-01", until="2015-12-31", use.backtest=TRUE)

## End(Not run)
```

# Index