

Package ‘portfolioSim’

April 19, 2009

Title Framework for simulating equity portfolio strategies

Version 0.2-5

Date 2009-02-06

Author Jeff Enos <jeff@kanecap.com> and David Kane <dave@kanecap.com>, with contributions from Kyle Campbell <Kyle.W.Campbell@williams.edu>

Description Classes that serve as a framework for designing equity portfolio simulations.

Maintainer Jeff Enos <jeff@kanecap.com>

Depends R (>= 2.4.0), methods, lattice, portfolio (>= 0.4-0)

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2009-02-07 14:22:10

R topics documented:

portfolioSim-package	2
instantData-class	3
loadIn	4
orderable-class	5
periodData-class	5
portfolioSim-class	6
saveOut	7
sdiDf-class	8
simData-class	9
simDataInterface-class	9
simResult-class	10
simResultSinglePeriod-class	11
simSummaryInterface-class	11
simTrades-class	12

simTradesInterface-class	13
starmine.sim	13
stiFromSignal-class	14
stiPresetTrades-class	15

Index	17
--------------	-----------

portfolioSim-package

Framework for simulating equity portfolio strategies

Description

Classes that serve as a framework for designing equity portfolio simulations.

Details

Package: portfolioSim
 Version: 0.2-5
 Date: 2009-02-06
 Depends: R (>= 2.4.0), methods, lattice, portfolio (>= 0.4-0)
 License: GPL (>= 2)
 LazyLoad: yes

Index:

instantData-class	Class "instantData"
loadIn	Load data from various formats.
orderable-class	Class "orderable"
periodData-class	Class "periodData"
portfolioSim-class	Class "portfolioSim"
portfolioSim-package	Framework for simulating equity portfolio strategies
saveOut	Save data in various formats.
sdiDf-class	Class "sdiDf"
simData-class	Class "simData"
simDataInterface-class	Class "simDataInterface"
simResult-class	Class "simResult"
simResultSinglePeriod-class	Class "simResultSinglePeriod"
simSummaryInterface-class	Class "simSummaryInterface"
simTrades-class	Class "simTrades"
simTradesInterface-class	Class "simTradesInterface"
starmine.sim	StarMine Rankings, 1995, and supplementary

```

data.
stiFromSignal-class      Class "stiFromSignal"
stiPresetTrades-class    Class "stiPresetTrades"

```

Further information is available in the following vignettes:

`portfolioSim` Performing equity investment simulations with the `portfolioSim` package (source, pdf)

Author(s)

Jeff Enos <jeff@kanecap.com> and David Kane <dave@kanecap.com>, with contributions from Kyle Campbell <Kyle.W.Campbell@williams.edu>

Maintainer: Jeff Enos <jeff@kanecap.com>

```
instantData-class  Class "instantData"
```

Description

Contains cross-sectional simulation data that pertains to a single instant in time, such as held positions and exposures.

Objects from the Class

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

Slots

instant: Object of class "orderable" specifying the instant to which the object pertains.

equity.long: Object of class "numeric" containing the total market value of held long positions at this instant.

equity.short: Object of class "numeric" containing the total market value of held short positions at this instant.

size.long: Object of class "numeric" containing the total number of held long positions at this instant.

size.short: Object of class "numeric" containing the total number of held short positions at this instant.

holdings: Object of class "portfolio" reflecting the portfolio held at this instant in the simulation.

exposure: Object of class "exposure" containing exposures of the `holdings` portfolio to a set of factors at this instant.

Methods

saveOut signature(object = "instantData", type = "character", fmt = "missing", out.loc = "character", name = "character", verbose = "logical"): save this object. Currently only one format, binary .RData, is available, and so the `fmt` parameter is missing here.

Author(s)

Jeff Enos <jeff@kanecap.com>

See Also

[periodData-class](#)

loadIn *Load data from various formats.*

Description

Generic function for loading data into an object.

Usage

```
loadIn(object, in.loc, fmt, ...)
```

Arguments

<code>object</code>	Object to be populated with data.
<code>in.loc</code>	Location or origin of the data.
<code>fmt</code>	Format in which the data is stored.
<code>...</code>	Any other parameters needed by the implementing method.

Value

The object with data loaded from `in.loc`, usually.

Author(s)

Jeff Enos <jeff@kanecap.com>

See Also

[saveOut](#)

orderable-class *Class "orderable"*

Description

A class union of classes that can be used to order observations, in particular the periods of a simulation. Currently, this includes `numeric`, `character`, `logical`, `POSIXt`, and `Date`.

Objects from the Class

A virtual Class: No objects may be created from it.

Author(s)

Jeff Enos <jeff@kanecap.com>

periodData-class *Class "periodData"*

Description

Contains data from the simulation that pertains to the passing of time during a period, such as performance and trading activity.

Objects from the Class

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

Slots

period: Object of class `"orderable"` specifying the period to which the object pertains.

turnover: Object of class `"numeric"` that contains the total value of trades, in a reference currency, executed during the trading period.

universe.turnover: Object of class `"numeric"` that contains the total value of positions, in a reference currency, that have exited the universe. Positions in securities held in the previous period but delisted in this period would contribute to this value.

performance: Object of class `"performance"` specifying return and profit for the period.

contribution: Object of class `"contribution"` specifying contributions of various categories of positions to total performance for the period.

trades: Object of class `"trades"` specifying the set of trades executed during this period.

Methods

saveOut signature(object = "periodData", type = "character", fmt = "missing", out.loc = "character", name = "character", verbose = "logical"): save this object. Currently only one format, binary .RData, is available, and so the `fmt` parameter is missing here.

Author(s)

Jeff Enos (jeff@kanecap.com)

See Also

[instantData-class](#)

portfolioSim-class *Class "portfolioSim"*

Description

When beginning a new simulation, the first step is to construct an object of class `portfolioSim` which will contain all the information required by the simulator. An instance of class `portfolioSim` represents a unique simulation, which can then be run at any time by calling the `runSim` method.

Objects from the Class

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

Slots

periods: A data frame listing the periods to be used in the simulation. Each period represents a single iteration of the simulator, in which a new set of trades is calculated and carried out. The periods data frame must have columns `period`, `start`, and `end`. The `period` column contains labels which are used throughout the simulator to represent each period. The `start` and `end` columns are used to differentiate between saved data from before and after the trades are performed in each period. Generally, these columns should contain the actual dates corresponding to each period.

freq: The annual frequency of the periods listed in the periods slot. For example, the frequency corresponding to the periods data frame shown above is 4. When running a simulation with monthly periods, the frequency should be 12. With daily periods, it should be 252, the total number of trading days in a year.

trades.interface: A trades interface object of some class containing the virtual class `simTradesInterface`. The trades interface represents the implementation of the trading strategy to be tested in the simulation. Based on the current portfolio and the data available for a given period, the trades interface contains some mechanism for determining a set of trades to make. These trades are encapsulated in a `simTrades` object which the interface returns to the simulator.

data.interface: A data interface object of some class containing the virtual class `simDataInterface`. The data interface serves to transform the raw data used in the simulation into an object of class `simData`, containing information on a single period.

summary.interface: An optional summary interface object of a class containing the virtual class `simSummaryInterface`. The summary interface allows the user to specify information to be saved out during the simulation beyond that supported by the result classes `instantData` and `periodData`.

start.holdings: A portfolio object representing the portfolio at the start of the simulation. If this slot is not specified, the simulator starts with an empty portfolio. See the documentation in the `portfolio` package for information on constructing a portfolio.

fill.volume.pct: Object of class "numeric" describing the maximum percentage of the daily trading volume of a stock that the simulator is allowed to trade in a single period. The default is 15. If set to `Inf`, all trades produced by the trades interface will be done, regardless of whether some of the associated securities are absent in sim data or have an NA market data for the period.

exp.var: An object of class "character" listing additional variables to be used when analyzing the exposures for each period.

contrib.var: Object of class "character" listing additional variables to be used when analyzing the contributions for each period.

out.loc: Object of class "character" describing the location at which to save the results of the simulation.

out.type: Object of class "character" listing the types of data to be saved out.

Methods

initialize signature(.Object = "portfolioSim"): Checks for and initializes preset type combinations.

runSim signature(object = "portfolioSim"): Run the simulation.

Author(s)

Jeff Enos (jeff@kanecap.com)

saveOut

Save data in various formats.

Description

Generic function for saving an object.

Usage

```
saveOut(object, type, fmt, out.loc, name, verbose, ...)
```

Arguments

<code>object</code>	Object to save.
<code>type</code>	Type of saving to perform, such as saving partial data from an object.
<code>fmt</code>	Format in which to save.
<code>out.loc</code>	Location in which to store the data.
<code>name</code>	Name for the saved object.
<code>verbose</code>	Whether the saving process should display informative output.
<code>...</code>	Any other parameters needed by the implementing method.

Value

The object just saved, usually.

Author(s)

Jeff Enos (jeff@kanecap.com)

See Also

[loadIn](#)

sdiDf-class

Class "sdiDf"

Description

Class "sdiDf" is an interface for converting an object of class "data.frame" into an object of class "simData".

Objects from the Class

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

Slots

data: Object of class "data.frame" storing the data to be used in the simulation.

period.var: Object of class "character" representing the column in data containing some object of class "orderable" to be used as periods in the simulation.

Extends

Class "simDataInterface", directly.

Methods

getSimData signature(object = "sdiDf", period = "orderable", verbose = "logical"): Returns an object of class "simData" containing the data for only that period specified by period.

Author(s)

Jeff Enos <jeff@kanecap.com>

simData-class *Class "simData"*

Description

Class "simData" stores the data to be used in the simulation.

Objects from the Class

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

Slots

data: Object of class "data.frameOrNull" with columns "period", "id", "start.price", "end.price", and "ret".

Author(s)

Jeff Enos <jeff@kanecap.com>

simDataInterface-class *Class "simDataInterface"*

Description

Virtual class that must be extended to provide data for the periods of a simulation.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

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

Author(s)

Jeff Enos <jeff@kanecap.com>

simResult-class *Class "simResult"*

Description

The highest level object that contains simulation result data.

Objects from the Class

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

Slots

freq: Object of class "numeric" that specifies the frequency of periods in the simulation with respect to a year. A value of 1 indicates annual periods, 12 monthly periods, etc.

data: Object of class "list" that contains `singlePeriodResult` objects for each period.

errors: Object of class "list" that contains a record of any errors caught during the processing of each period.

type: Object of class "character" that specifies the type of data contained in this object. Usually pertains to partial saving of period and instant information.

summary.interface: Object of class "simSummaryInterfaceOrNull" that can be used as an additional saving filter for single-period results.

Methods

loadIn signature(object = "simResult", in.loc = "character", fmt = "missing"): load in the simulation data stored in `in.loc`. Currently only one format, binary `.RData`, is available, and so the `fmt` parameter is missing here.

plot signature(x = "simResult", y = "missing"): plot simulation results.

saveOut signature(object = "simResult", type = "missing", fmt = "missing", out.loc = "character", name = "missing", verbose = "logical"): save this object. Currently only one format, binary `.RData`, is available, and so the `fmt` parameter is missing here.

summary signature(object = "simResult"): summarize the simulation.

Author(s)

Jeff Enos (jeff@kanecap.com)

```
simResultSinglePeriod-class
      Class "simResultSinglePeriod"
```

Description

Contains simulation result data for a single period.

Objects from the Class

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

Slots

start.data: Object of class "instantData" that contains cross-sectional data as of the start of the period.

end.data: Object of class "instantData" that contains cross-sectional data as of the end of the period.

period.data: Object of class "periodData" that contains data for the period involving the passage of time.

Methods

loadIn signature(object = "simResultSinglePeriod", in.loc = "character", fmt = "missing"): load in the simulation data stored in in.loc. Currently only one format, binary .RData, is available, and so the fmt parameter is missing here.

saveOut signature(object = "simResultSinglePeriod", type = "character", fmt = "missing", out.loc = "character", name = "missing", verbose = "logical"): save this object. Currently only one format, binary .RData, is available, and so the fmt parameter is missing here.

Author(s)

Jeff Enos (jeff@kanecap.com)

```
simSummaryInterface-class
      Class "simSummaryInterface"
```

Description

The summary interface allows the user to specify information to be saved out during the simulation beyond that supported by the result classes `instantData` and `periodData`.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class "simSummaryInterfaceOrNull", directly.

Methods

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

Author(s)

Kyle Cambell <Kyle.W.Campbell@williams.edu>

simTrades-class *Class "simTrades"*

Description

Class "simTrades" stores a list of trades to be made in a single period.

Objects from the Class

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

Slots

period: Object of class "orderable" representing a single period for which these trades should be performed.

trades: Object of class "trades" containing the trades to be performed during this period.

Author(s)

Jeff Enos <jeff@kanecap.com>

```
simTradesInterface-class
      Class "simTradesInterface"
```

Description

Virtual class that must be extended to provide the simulator with the set of trades to execute during each period. The trades returned by the interface represent the implementation of the trading strategy tested in the simulation.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

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

Author(s)

Jeff Enos (jeff@kanecap.com)

```
starmine.sim      StarMine Rankings, 1995, and supplementary data.
```

Description

StarMine rankings of some stocks in 1995, with the minimal set of supplementary data required for running a simulation.

Usage

```
data(starmine.sim)
```

Format

A data frame with 53328 observations on the following 14 variables.

date Date on which the observation was recorded. The dates have a monthly frequency. Dates range from 1995-01-31 to 1995-11-30.

id Unique identifier for each stock.

name Full company name.

country Country of the exchange on which the company is listed.

sector Sector to which the stock belongs.

cap.usd Market capitalisation of the company in USD.

size cap.usd normalized to $N(0,1)$.

smi StarMine Indicator (smi) score for each security and date if a score was issued.

fwd.ret.1m 1 month forward return.

fwd.ret.6m 6 month forward return.

price.usd Adjusted price, in USD, of the security at the end of the period specified by `date`.

prior.close.usd Adjusted price, in USD, of the security at the end of the period prior to the period specified by `date`.

volume Adjusted volume of the security on the last day of the period specified by `date`.

ret.1m Total return for the period (month) specified by `date`.

Source

StarMine Corporation. For more information, see <http://www.starmine.com>.

Examples

```
data(starmine.sim)
head(starmine.sim)
```

```
stiFromSignal-class
      Class "stiFromSignal"
```

Description

Class "stiFromSignal" is an interface that stores information regarding portfolio formation and trading to be used in determining trades during the simulation.

Objects from the Class

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

Slots

in.var: Object of class "character" representing a column in the data interface to be used the "in.var" for creating portfolios.

type: Object of class "character" representing the type of weight calculation to be used.

size: Object of class "characterOrNumeric" representing the size of the portfolio to be created during the simulation.

sides: Object of class "character" containing "long", "short", or both, indicating the type of portfolio to be created.

equity: Object of class "numeric" representing the equity for the portfolio.

target: Object of class "environment" representing the environment in which to search for the target portfolio.

rebal.on: Object of class "orderable" containing the periods at which the portfolio should be rebalanced during the simulation.

trading.style: Object of class "character" representing the trading style to use during the simulation. Defaults to "immediate".

Extends

Class "simTradesInterface", directly.

Methods

initialize signature(.Object = "stiFromSignal"): Initializes the interface by setting the target environment.

getSimTrades signature(object = "stiFromSignal", period = "orderable", holdings = "portfolio", sim.data = "simData", verbose = "logical"): Returns an object of class "simTrades" containing all the trades that should be made for this period.

Author(s)

Jeff Enos (jeff@kanecap.com)

```
stiPresetTrades-class
```

```
Class "stiPresetTrades"
```

Description

A trades interface that provides a predetermined set of trades for instructional purposes.

Objects from the Class

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

Slots

periods: Object of class "orderable" containing the periods for which trades are available.

sim.trades: Object of class "list" containing the trades for each period.

Extends

Class "simTradesInterface", directly.

Methods

getSimTrades signature(object = "stiPresetTrades", period = "orderable", holdings = "portfolio", sim.data = "simData", verbose = "logical"): get the trades for period.

Author(s)

Kyle Cambell <Kyle.W.Campbell@williams.edu>

Index

*Topic **classes**

instantData-class, 3
orderable-class, 4
periodData-class, 5
portfolioSim-class, 6
sdiDf-class, 8
simData-class, 9
simDataInterface-class, 9
simResult-class, 10
simResultSinglePeriod-class, 11
simSummaryInterface-class, 11
simTrades-class, 12
simTradesInterface-class, 13
stiFromSignal-class, 14
stiPresetTrades-class, 15

*Topic **datasets**

starmine.sim, 13

*Topic **methods**

loadIn, 4
saveOut, 7

*Topic **package**

portfolioSim-package, 1

Date-class (*orderable-class*), 4

getSimData (*sdiDf-class*), 8

getSimData, sdiDf, orderable, logical-method (*sdiDf-class*), 8

getSimTrades (*stiFromSignal-class*), 14

getSimTrades, stiFromSignal, orderable, portfolioSim, logical-method (*stiFromSignal-class*), 14

getSimTrades, stiPresetTrades, orderable, portfolioSim, logical-method (*stiPresetTrades-class*), 15

initialize, portfolioSim-method (*portfolioSim-class*), 6

initialize, stiFromSignal-method (*stiFromSignal-class*), 14

instantData-class, 5

instantData-class, 3

loadIn, 4, 8

loadIn, simResult, character, missing-method (*simResult-class*), 10

loadIn, simResultSinglePeriod, character, missing-method (*simResultSinglePeriod-class*), 11

orderable-class, 4

periodData-class, 3

periodData-class, 5

plot, simResult, missing-method (*simResult-class*), 10

portfolioSim (*portfolioSim-package*), 1

portfolioSim-class, 6

portfolioSim-package, 1

runSim (*portfolioSim-class*), 6

runSim, portfolioSim, logical-method (*portfolioSim-class*), 6

saveOut, 4, 7

saveOut, instantData, character, missing, character, (*instantData-class*), 3

saveOut, periodData, character, missing, character, (*periodData-class*), 5

saveOut, simResult, missing, missing, character, miss (*simResult-class*), 10

saveOut, simResultSinglePeriod, character, missing, (*simResultSinglePeriod-class*), 11

sdiDf-class, 8

simData-class, 9

simDataInterface-class, 9

simResult-class, 10

simResultSinglePeriod-class, 11

simSummaryInterface-class, 11

simSummaryInterfaceOrNull-class
 (*simSummaryInterface-class*),
 11

simTrades-class, 12

simTradesInterface-class, 13

starmine.sim, 13

stiFromSignal-class, 14

stiPresetTrades-class, 15

summary, simResult-method
 (*simResult-class*), 10