Package 'backtestGraphics'

July 22, 2025

Type Package

Title Interactive Graphics for Portfolio Data

Description Creates an interactive graphics interface to visualize backtest results of different financial instruments, such as equities, futures, and credit default swaps. The package does not run backtests on the given data set but displays a graphical explanation of the backtest results. Users can look at backtest graphics for different instruments, investment strategies, and portfolios. Summary statistics of different portfolio holdings are shown in the left panel, and interactive plots of profit and loss (P&L), net market value (NMV) and gross market value (GMV) are displayed in the right panel.

Version 0.1.8

Date 2025-01-02

License GPL-3

LazyData true

Depends R (>= 3.5.0)

Imports dygraphs, dplyr, scales, xts, shiny, tibble

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-08 05:40:13 UTC

RoxygenNote 7.3.2

Encoding UTF-8

Author Yanrong Song [aut, cre],
Zijie Zhu [aut],
David Kane [aut],
Ziqi Lu [aut],
Karan Tibrewal [aut],
Fan Zhang [aut]

Maintainer Yanrong Song <yrsong129@gmail.com>

Contents

`backtestGraphics-package`

A package to visualize backtest results

Description

Package	backtestGraphics
Type:	Package
Version:	0.1-1
Date:	2015-07-14
License:	GPL-3

Details

The backtestGraphics package creates an interactive graphics interface for commodity futures portfolios, equity portfolios and credit default swap. The interface contains three drop-down menus that allow the user to look at different portfolios, different investment strategies and different sectors or commodities. Summary statistics of the start date, end date, allocated capital, average GMV, number of instruments, cumulative P&L, annualized P&L, annualized volatility, sharpe ratio, best

month and worst month are displayed in the summary screen. More details of the top three drawdowns, three best performers and three worst performers are displayed in another detail screen. An interactive plot for the cumulative P&L or the daily P&L is shown at the top and another interactive graph for nmv, gmv and number of contracts is displayed at the bottom.

The package contains a main function `backtestGraphics` that takes in a data frame of backtest results, and returns summary statistics about the data frame and plots the historical traces of market values and profits. The package also contains many helper functions that perform the calculations and plotting for the `backtestGraphics` function. These helper functions can only be called within the `backtestGraphics` function.

The package contains three data frames, `commodity`, `equity` and `credit`. The `commodity` data frame contains the backtest results for 28 commodities in the futures market. The `equity` data frame contains the backtest results for random 50 stocks. The `credit` data frame is the backtest result for credit default swap. The user can simply call `backtestGraphics` with these data frames. An example will be `backtestGraphics(x = commodity)`.

In order to use the user's own data frame, sometimes the user might need to specify the column names of her data frame and pass them into the function. Type `?backtestGraphics` for more details.

Author(s)

Maintainer: Yanrong Song <yrsong129@gmail.com>

Authors:

- Zijie Zhu <zijie.miller.zhu@gmail.com>
- David Kane <dave.kane@gmail.com>
- Ziqi Lu <ziqi.lu@williams.edu>
- Karan Tibrewal <karan.tibrewal@williams.edu>
- Fan Zhang <fan.zhang@williams.edu>

`backtestGraphics` *Interactive Backtest Graphics*

Description

This function takes a data frame and returns an interactive interface for the backtest data. The interface contains drop-down menus for different strategies, portfolios and instruments/sectors. The user can slice her data set according to the strategies for selecting instruments or the portfolio numbers of her individual portfolios. The user can also look at a specific group of instruments according to the sector of instruments. These sectors will be in the same dropdown menu as instruments. The elements in each of the three dropdown menus are generated according to the input data set. The dropdown menus for strategies and portfolio numbers will only contain "Strategy Summary" or "Portfolio Summary" if there's no strategy or portfolio information available in the input data set.

Usage

```
backtestGraphics(
  x,
  trade.freq = NULL,
  name.var = "name",
  id.var = "id",
  date.var = "date",
  nmv.var = "nmv",
  gmv.var = "gmv",
  pnl.var = "pnl",
  contract.var = "contract",
  capital.num = NULL,
  sector.var = "sector",
  strategy.var = "strategy",
  substrategy.var = "substrategy",
  portfolio.var = "portfolio"
)
```

Arguments

x	is a data frame that contains the necessary information.
trade.freq	is the trading frequency of the data set, numeric type. The variable uses the number of dates between two trading dates to indicate trading frequency. The default value of this variable is NULL, and in this case the function will automatically calculate the trading period.
name.var	is the column name of the instrument name column in the input data set, character type. The default value of this variable is "name". The user has to specify name.var if she passes in a data frame with a different column name for the instrument name. The input data set must contain either a name column or an ID column.
id.var	is the column name of the instrument ID column, character type. The default value of this variable is "id". If the input data set labels the column of instrument ID's with some other column name, the user has to pass in the column name here. The input data set has to contain either a name column or an ID column.
date.var	is the column name of the date column, character type. The default value of this variable is "date". This column has to exist, and the column name has to be correct in order for the function to process the data set properly.
nmv.var	is the column name of the "net market value" column, character type. The default value of this variable is "nmv". The input data set has to contain either a "net market value" column or a "number of contract" column.
gmv.var	The column name of the "gross market value" column if exists, character type. The default value of this variable is "gmv". Such column will be automatically calculated from the "net market value" column if it does not exist.
pnl.var	The column name of the profit-and-loss column, character type. The default value of this variable is "pnl". The data set has to contain such column so that the function can function properly. If such column is missing, the function will return an error indicating the problem.

`contract.var` The column name of the contract number column, character type. The default value of this variable is "contract". If such column is missing, the function will instead use net market value as the contract number for each day.

`capital.num` The constant number of allocated capital for the whole portfolio, numeric type. The default value of such variable is `NULL`. The function will use the number of allocated capital to calculate return rates. If the allocated capital for the portfolio is not specified, the function will use each day's gross market value to calculate the return rate for each day.

`sector.var` is the column name of the sector column, character type. The default value of this variable is "sector". The sector column helps the user to group instruments into big groups according to industries. The function will still perform properly if the column name for sector is wrong.

`strategy.var` The column name of the strategy column, if any. Character type. The default value of this variable is "strategy". This column can be missing from the input data set.

`substrategy.var`
The column name of the substrategy column, if any. Character type. The default value of this variable is "substrategy". This column can be missing from the input data set.

`portfolio.var` The column name of the portfolio number column, if any. Character type. The default value of this variable is "portfolio". This column can be missing from the input data set.

Details

The user can also slice the data set according to the division of the data set into main strategies, and the division of any main strategy into small substrategies. The "substrategy" column of the data set should indicate the division of main strategies, if there are any substrategies. Note that the structure of substrategies should conform to main strategies. That is, a single substrategy should be under only one main strategy.

Summary statistics are displayed in tables under the drop-down menu. The summary statistics contain data for the backtesting time horizon, profits, returns, volatility, market values, best/worst performers and biggest drawdowns. Different interactive plots for the cumulative P&L, daily P&L, Net Market Value (NMV), Gross Market Value (GMV) and number of contracts are shown in the right panel.

The `backtestGraphics` function takes in a data set as well as the names of essential columns in the data set, if these names are different from the default ones. The column names should be assigned to the corresponding variables so that the function can recognize these columns. If some essential columns are missing, the function will try to fill in these columns with existing data. If the function fails to do so due to a lack of data, the function will return an error about the missing columns. If a column name is wrong, the function will treat that column as a missing column. The input data set has to contain either a name column or an ID column, a date column, either a net market value column or a number of contract column and a P&L column. The other columns either help the user slice the data set into different pieces to diagnose the data better, or can be calculated from the data inside the input data set.

Value

a Shiny interface. The interactive shiny interface displays P&L and cumulative P&L on one chart, and number of contracts, net market value and gross market value on the other chart. Some summary statistics about return and performances are displayed on the left sidebar as tables.

Examples

```
if(interactive()) {backtestGraphics(credit)}
```

---

best_worst_month *Find the best-performing and the worst-performing months*

Description

This function takes in a data set and returns a list with the best and the worst month and their respective pnl's.

Usage

```
best_worst_month(x)
```

Arguments

x	A data frame that contains data for individual instruments.

Value

A list with the best month and the worst month, as well as their respective p&l's.

---

best_worst_three *Find the three best-performing and worst-performing commodities*

Description

This function takes in a data set and returns the best three and worst three commodity names and their respective pnls. All these data will be presented as a formatted table.

Usage

```
best_worst_three(x)
```

Arguments

x	A data frame that contains data for individual commodities.

Value

output A list of the best three and worst three performers with their commodity names, ids and respective pnls

---

`calc_pnl` *Calculate cumulative pnl, mean pnl, dollar sharpe ratio*

Description

This function takes in a data frame and calculates the cumulative P&L, daily P&L, annualized P&L, P&L volatility and dollar sharpe ratio as well as three biggest drawdowns. There are also average daily return rate and the standard deviation of daily return rates.

Usage

```
calc_pnl(x, trade.freq = 7)
```

Arguments

`x`	A data frame that contains data for individual commodities.
`trade.freq`	The trading frequency of the portfolio, numeric

Value

a list ith cumulative pnl, mean pnl, annualized volatility of pnl, dollar sharpe ratio and drawdown

---

`cleanup_column` *Clean up the input data set*

Description

This function cleans up the input data set in the backtestGraphics function. This function checks if all the columns required by the backtestGraphics function exist in the input data set. If any columns are missing, this function will try to calculate or fill in the missing columns. If the input data set misses too much data/columns, this function will return an error indicating the missing columns.

Usage

```
cleanup_column(
  x,
  name.var,
  id.var,
  date.var,
  nmv.var,
  gmv.var,
  pnl.var,
  contract.var,
  sector.var,
  strategy.var,
  substrategy.var,
  portfolio.var
)
```

Arguments

x	is a data frame that contains the necessary information.
name.var	is the column name of the instrument name column in the input data set, character type. The default value of this variable is "name". The user has to specify name.var if she passes in a data frame with a different column name for the instrument name. The input data set must contain either a name column or an ID column.
id.var	is the column name of the instrument ID column, character type. The default value of this variable is "id". If the input data set labels the column of instrument ID's with some other column name, the user has to pass in the column name here. The input data set has to contain either a name column or an ID column.
date.var	is the column name of the date column, character type. The default value of this variable is "date". This column has to exist, and the column name has to be correct in order for the function to process the data set properly.
nmv.var	is the column name of the "net market value" column, character type. The default value of this variable is "start.nmv". The input data set has to contain either a "net market value" column or a "number of contract" column.
gmv.var	The column name of the "gross market value" column if exists, character type. The default value of this variable is "gmv". Such column will be automatically calculated from the "net market value" column if it does not exist.
pnl.var	The column name of the profit-and-loss column, character type. The default value of this variable is "pnl.adj". The data set has to contain such column so that the function can function properly. If such column is missing, the function will return an error indicating the problem.
contract.var	The column name of the contract number column, character type. The default value of this variable is "num.contract.start". If such column is missing, the function will instead use net market value as the contract number for each day.
sector.var	is the column name of the sector column, character type. The default value of this variable is "sector". The sector column helps the user to group instruments

into big groups according to industries. The function will still perform properly if the column name for sector is wrong.

`strategy.var` The column name of the strategy column, if any. Character type. The default value of this variable is "strategy". This column can be missing from the input data set.

`substrategy.var` The column name of the substrategy column, if any. Character type. The default value of this variable is "substrategy". This column can be missing from the input data set.

`portfolio.var` The column name of the portfolio number column, if any. Character type. The default value of this variable is "portfolio". This column can be missing from the input data set.

Value

x A data frame that only contains the needed data. The column names are cleaned up here.

`commodity` *Commodity Futures Data from 2003 to 2005.*

Description

This data frame contains the futures backtest results of 28 commodities from 2003-01-01 to 2005-12-30. The data frame contains daily positions and profits for each commodity.

Format

A data frame with 11 variables

- name = The name of each commodity.
- id = The specific ID for each commodity.
- date = The individual trading date.
- sector = The sector a commodity.
- portfolio = The portfolio number the holding belongs to.
- strategy = The strategy under which substrategies and portfolios are established. The same commodity can belong to different strategies at the same time.
- substrategy = The substrategy under which portfolios are established. Substrategy is only a finer division of strategies.
- gmv = The gross market value of the commodity on the trading date.
- nmv = The net market value of the commodity on the trading date.
- pnl = The adjusted P&L of a commodity on the trading date.
- contract = The number of contracts of a commodity on that trading date.

Details

The commodity data frame contains three layers of complexity. The first layer is the strategy layer. The `strategy` column contains three different strategies that divide the whole data frame into three part. Each strategy is further divided into different substrategies, and all these substrategies are contained in the `substrategy` column. For each substrategy, different portfolios are formed regularly and overlapping with each other, so each substrategy is divided into different overlapping portfolios, contained in the `portfolio` column.

`create_instrument_optGroups`

Create opt group list for selectizeInput for instruments

Description

This function accepts the sector and id columns of the data set and create a list of groups as required by the selectizeInput optgroup functionality

Usage

```
create_instrument_optGroups(sector, id)
```

Arguments

`sector`	sector column
`id`	id column

Value

a list of groups for sector/ id as required by the selectizeInput optgroup functionality

---

`create_strategy_optGroups`

Create opt group list for selectizeInput for strategies

Description

This function accepts the strategy and substrategy columns of the data set and create a list of groups as required by the selectizeInput optgroup functionality

Usage

```
create_strategy_optGroups(strategy, substrategy)
```

Arguments

strategy	strategy column
substrategy	substrategy column

Details

We assume that substrategies pertaining to a given strategy uses the name of the strategy followed by a period and then the substrategy number. For example, substrategy 2 for strategy 1 must be named "Strategy 1.2".

Value

a list of groups for strategy/ substrategy as required by the selectizeInput optgroup functionality

---

credit *Credit Default Swap Data from 2007 to 2009.*

Description

This data frame contains the information of 260 credit default swaps (CDS) from 2007-01-02 to 2009-12-31. The data sets is actually a combination of CDS backtest conducted under daily, weekly, monthly, and quartrly trading frequency. The strategy column contains the trading frequency.

Format

A data frame with 7 variables

- name = The name of each credit default swap (CDS).
- date = The trading date.
- sector = The sector the CDS belongs to.
- strategy = The trading strategy of the credit default swap, including "daily", "weekly", "monthly" and "quarterly".
- gmv = The gross market value of the CDS held on that day.
- nmv = The net market value of the CDS held on that day.
- pnl = The P&L value (adjusted) of the CDS on that day.

cumpnl_plot *Draw an interactive line plot for cumulative P&L*

Description

This function takes a data frame and returns an interactive plot for the cumulative P&L of the portfolio data. The plot is zoomable and specific data can be shown once the cursor is placed on the graph.

Usage

```
cumpnl_plot(x)
```

Arguments

x	A data frame that the function takes in.

Value

A plot. Red bars indicate loss and green bars indicate profit.

drawdown *The Three Biggest Drawdowns in the portfolio*

Description

Show the top 3 drawdowns including start and end dates, as well as decrease in returns during the drawdown period. All the information will be returned in a table, with all the numbers properly formatted.

Usage

```
drawdown(x)
```

Arguments

x	A data frame with date and return columns

Details

If the data set is not big enough that there are fewer drawdowns than required by the user, the function will throw NA's into the table so that the table will still contain as many rows as the user demands.

Value

A data frame that contains the starting date, end date and values of the three biggest drawdowns.

equity *Equity Data from 2005 to 20014.*

Description

This data frame contains the information of 50 equities from 2005-05-02 to 2014-04-30. These 50 stocks are randomly chosen from a backtest results of the whole stock markets.

Format

A data frame with 5 variables

- name = The name of this equity.
- date = trading date.
- sector = The sector this equity belongs to. There are nine sectors in total.
- nmv = The net market value of the equity held on that day.
- pnl = The adjusted P&L of the equity on that day. P&L is adjusted according to the bid offer costs.

interactive_plot *Interactive plot with dygraph*

Description

This function takes a data frame and returns an interactive plot for the portfolio data. The plot is zoomable and specific data can be shown once the cursor is placed on the graph.

Usage

```
interactive_plot(x, type)
```

Arguments

x	A data frame that the function takes in.
type	The value depicted by the plot, choices are nmv, gmv, contract, pnl and ret.

Details

For all bar plots, green represents profit and red represents loss. This function also calculates gmv, cumulative P&L and return rates if these statistics are not in the data frame. Number of contracts will all be zero if the column is missing.

Value

A plot. Red bars indicate loss and green bars indicate profits.

next_trading_day *Next trading day*

Description

This function gets the next trading day as defined by is.trading.day()

Usage

```
next_trading_day(d)
```

Arguments

d	Date relative to which to get the next trading day.

Value

Date The next trading day of date d.

slice_data *Slice Data Set*

Description

This function is part of the Shiny reactive functions. The function looks at what strategy, portfolio and instrument the user selects, and then slices the data set into different pieces or summarizes the data set into a summary.

Usage

```
slice_data(x, input, capital.num)
```

Arguments

x	is an input data frame that is to be sliced according to the user's selection of strategy, portfolio and instrument.
input	is a Shiny object that contains all the shiny inputs as a list.
capital.num	is the amount of allocated capital for the portfolio, numeric type. The function will use capital.num to calculate return rates if available, or use everyday's GMV to calculate return rates instead.

Value

A small data set that has been retrieved from the previous data set.

stat_calculation *Calculate Statistics*

Description

This helper function calculates the summary of the input data set. The function takes in the sliced data set and the calculated trading frequency, and calculate statistics about the market value, returns and volatility for later display.

Usage

```
stat_calculation(x.list)
```

Arguments

x.list	is the input data set. Such data set has been sliced by slice_data function, and is labeled as "intermediate data set".

Value

f A list that contains all calculated numbers.

sum_data *Summarize data for the overall portfolio*

Description

This function takes in a data set and returns the summary for each day's net market value, profit and loss, contract number and return rate.

Usage

```
sum_data(x, sector.selected = NULL, capital.num = NULL)
```

Arguments

x	A data frame that contains data for individual commodities.
sector.selected	The sector that the function is doing summarization for. When "sector.selected = NULL", the function performs summarization across the whole data set.
capital.num	The constant number of allocated capital to calculate return, number. Default value is NULL. When "capital.num = NULL", the function uses gross market value to calculate return rates for each row.

Value

A data frame that summarizes the market values and returns across all commodities.

`trade_freq` *Find Trading Frequency*

Description

This helper function looks for the trading frequency of the input data set. It can identify daily, weekly, monthly or yearly trading frequency in the data set.

Usage

```
trade_freq(x)
```

Arguments

`x`	The input data set whose trading period we are interested in

Value

trade.freq The number that indicates the trading frequency.

Index