Package 'TradeStrategyAnalyzer'

June 11, 2011

Type Package

Title Trade Strategy Analyzer

Version 1.0

Date 2011-06-10

Author Winnie Cheng, Tirto Adji

Maintainer Winnie Cheng <wwcheng@alum.mit.edu>

Description Simulates and visualizes different trading strategies on market data

License GPL (>= 2)

Depends DBI, RSQLite, stats, googleVis, RJSONIO, ggplot2, digest

LazyLoad yes

R topics documented:

TradeStrategyAnalyzer-package . . . 2
AddPortfolio . . . 3
BuyFee . . . 3
calcSumHoldings . . . 4
calendarHeatMap . . . 4
cashTicker . . . 5
DetermineOrders_MeanRevert . . . 5
DetermineOrders_RandomBuyAndHold . . . 6
GetPortfolioPerformance . . . 6
GetPortfolioWorthAtTime . . . 7
gvisAnnotatedTimeLine2 . . . 8
gvisMotionChart2 . . . 9
HoldingsInfo . . . 10
LoadMarketToDB . . . 10
LoadStocksToDB . . . 11
MeanRevertCorrelatePairwise . . . 11
portfolioBoxPlot . . . 12
portfolioHoldingSummaryChart . . . 13
PortfoliosInfo . . . 13
portfolioStocksChart . . . 14

PrintSimulatorState 14
readFromDB 15
schemaHoldingsTableName 15
schemaMarketTableName 16
schemaPortfoliosTableName 16
schemaStocksTableName 16
schemaTransactionsTableName 17
SellFee 17
SimulateBroker 17
stockCorrMatrixChart 18
stockCorrPairsCartesianChart 19
StoreCorrelationsToDB 19
StoreHoldingsToDB 20
StorePortfoliosToDB 20
StoreSimulationResults 21
StoreTransactionsToDB 21
TransactionsInfo 22
VisualizeCorrelation 22
writeToDb 23

Index **24**

`TradeStrategyAnalyzer-package`
Trade Strategy Analyzer

Description

This package contains a trade simulator, a collection of trading strategies and visualization techniques. For a description of the package, please refer to TradeStrategyAnalyzer_WhitePaper.pdf included in inst/doc.

Details

Package:	TradeStrategyAnalyzer
Type:	Package
Version:	1.0
Date:	2011-06-10
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Winnie Cheng <wwcheng@alum.mit.edu>, Tirto Adji <tirto_adji@yahoo.com>

References

Please refer to TradeStrategyAnalyzer_WhitePaper.pdf included in the package.

AddPortfolio *Creates a new portfolio*

Description

This function creates a new portfolio in the strategy simulator. Returns data frame of portfolios that includes this newly created portfolio.

Usage

```
AddPortfolio(PortfoliosInfo, portfolioId, algorithm, startAmount)
```

Arguments

PortfoliosInfo	Data frame for storing information on portfolios in simulator
portfolioId	Portfolio Id
algorithm	Name for this Portfolio
startAmount	Initial investment amount in dollars

Examples

```
## see TestSimulator.R
```

BuyFee *Transaction Fee for Buying a stock*

Description

This represents the value in dollars in transaction fee for purchasing a stock.

Format

To change the ticker: BuyFee <- 1

Examples

```
# To display current BuyFee
BuyFee
```

calcSumHoldings *Calculates Summary of Portfolio Holdings*

Description

Aggregates all of stocks value in portfolio holdings group by portfolio ID

Usage

```
calcSumHoldings(dbFile, histMarketData, holdingsData)
```

Arguments

dbFile	Database path and file name
histMarketData	Historical Market Data
holdingsData	Portfolio Holding Data

Examples

```
## see TestVisualizer.R
```

calendarHeatMap *Plots Calendar Heat Map*

Description

Plots Calendar Heat Map of a stock from our portfolio DB or from web services

Usage

```
calendarHeatMap(mData, ticker, startdate = "2009-08-21", enddate = "2010-08-20")
```

Arguments

mData	Market Data
ticker	Ticker symbol
startdate	start date
enddate	end date

References

Wickham, Hadley (2009) ggplot2 Chambers, J. (2008) Software for Data Analysis

Examples

```
## see TestVisualizer.R
```

cashTicker *Ticker Symbol for Cash*

Description

This is a ticker symbol used to represent cash in portfolio

Format

To change the ticker: cashTicker <- "CASH"

Examples

```
# To display current value for cashTicker
str(cashTicker)
```

DetermineOrders_MeanRevert *Generate Order Book using Mean Revert Trading Algorithm*

Description

This function generates a data frame of trade orders that result from running the mean revert trading algorithm.

Usage

```
DetermineOrders_MeanRevert(portfolioId, PortfoliosInfo, StockMarketDataset, numP
```

Arguments

portfolioId	Id of portfolio on which to apply mean revert algorithm
PortfoliosInfo	Data frame containing information on portfolios in simulator
StockMarketDataset	Data frame containing stock market dataset
numPairs	Number of trading pairs to be considered in the algorithm

Value

Returns a data frame containing trade orders. The attributes for each row / trade order are: "ACTIVITY_TS", "PORTFOLIOID", "TICKER", "NUMSHARES", "ORDERTYPE", "ORDERVALUE"

Examples

```
## See TestSimulator.R
```

`DetermineOrders_RandomBuyAndHold` *Generate Order Book using Random Buy-and-Hold Trading Algorithm*

Description

This function generates a data frame of trade orders that result from running the random buy-and-hold algorithm.

Usage

```
DetermineOrders_RandomBuyAndHold(portfolioId, PortfoliosInfo, StockMarketDataset
```

Arguments

`portfolioId`	Id of portfolio on which to apply mean revert algorithm
`PortfoliosInfo`	Data frame containing information on portfolios in simulator
`StockMarketDataset`	Data frame containing stock market dataset
`numCandidates`	Number of stocks to buy and hold

Value

Returns a data frame containing trade orders. The attributes for each row / trade order are: "ACTIVITY_TS", "PORTFOLIOID", "TICKER", "NUMSHARES", "ORDERTYPE", "ORDERVALUE"

Examples

```
## See TestSimulator.R
```

`GetPortfolioPerformance` *Computes Portfolio Return*

Description

This function computes the portfolio return based on the time period of the stock market dataset. It returns the profit/loss on the final asset value over the initial asset value.

Usage

```
GetPortfolioPerformance(portfolioId, PortfoliosInfo, HoldingsInfo, StockMarketDa
```

Arguments

`portfolioId`	Id of portfolio
`PortfoliosInfo`	Data frame containing information on portfolios in simulator
`HoldingsInfo`	Data frame contain information on asset holding for different portfolios in the simulator
`StockMarketDataset`	Data frame containing stock market dataset

Value

Overall Portfolio Return in Percentage

Examples

```
## See TestSimulator.R
```

`GetPortfolioWorthAtTime` *Computes the net value of a portfolio at a particular time*

Description

This function computes the net value of a portfolio at a particular instance in time.

Usage

```
GetPortfolioWorthAtTime(timestamp, portfolioId, StockMarketDataset, HoldingsInfo
```

Arguments

`timestamp`	Instance in time
`portfolioId`	Id of portfolio
`StockMarketDataset`	Data frame containing stock market dataset
`HoldingsInfo`	Data frame contain information on asset holding for different portfolios in the simulator

Value

Net value in dollars of the Portfolio at the specified time

Examples

```
## See TestSimulator.R
```

gvisAnnotatedTimeLine2 *modified version of gvisAnnotatedTimeLine2*

Description

has all of the functionalities of gvisAnnotatedTimeLine, plus optional chart title and description args and fixes to javascript api location

Usage

```
gvisAnnotatedTimeLine2(data,
                       datevar="",
                       numvar="",
                       idvar="",
                       titlevar="",
                       annotationvar="",
                       date.format="%Y/%m/%d",
                       options = list(),
                       chartid,
                       charttitle="",
                       chartdesc="")
```

Arguments

data	a data.frame. The data has to have at least two columns, one with date information (datevar) and one numerical variable.
datevar	column name of data which shows the date dimension.
numvar	column name of data which shows the values to be displayed
idvar	column name of data which identifies different groups of the data.
titlevar	column name of data which shows the title of the annotations.
annotationvar	column name of data which shows the annotation text.
date.format	specifies how the dates are reformatted to be used by JavaScript.
options	list of configuration options for Google Annotated Time Line.
chartid	character. If missing (default) a random chart id will be generated
charttitle	character. can be HTML snippets
chartdesc	character. can be HTML snippets

References

http://code.google.com/apis/visualization/documentation/gallery/motionchart.html

See Also

gvisAnnotatedTimeLine

Examples

```
# see TestVisualizer.R
```

gvisMotionChart2 *modified version of gvisMotionChart*

Description

has all of the functionalities of gvisMotionChart, plus optional chart title and description args

Usage

```
gvisMotionChart2(data,
                 idvar="id",
                 timevar="time",
                 date.format="%Y/%m/%d",
                 options = list(),
                 chartid,
                 charttitle="",
                 chartdesc="")
```

Arguments

data	a data.frame. The data has to have at least four columns with subject name (idvar), time (timevar) and two columns of numeric values. Further columns, numeric and character/factor are optional. The combination of idvar and timevar has to describe a unique row.
idvar	column name of data with the subject to be analysed.
timevar	column name of data which shows the time dimension.
date.format	specifies how the dates are reformatted to be used by JavaScript.
options	list of configuration options for Google Motion Chart.
chartid	character. If missing (default) a random chart id will be generated
charttitle	character. can be HTML snippets
chartdesc	character. can be HTML snippets

References

http://code.google.com/apis/visualization/documentation/gallery/motionchart.html

See Also

gvisMotionChart

Examples

```
## see TestVisualizer.R
```

HoldingsInfo *Data frame of asset holding for different portfolios*

Description

Data frame contain information on asset holding for different portfolios in the simulator

Format

A data frame with the following 4 variables.

`PORTFOLIOID` a numeric vector representing the Id of Portfolio

`ACTIVITY_TS` timestamp

`TICKER` stock ticker symbol

`NUMSHARES` number of shares held

Examples

```
## See TestSimulator.R
```

LoadMarketToDB *Loads stock market daily prices from csv file to R and database*

Description

This function reads in a .csv file containing individual daily stock prices into the R environment as well as storing the information in a database table.

Usage

```
LoadMarketToDB(schemaDbName, inputMarketInfoFile)
```

Arguments

`schemaDbName`	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
`inputMarketInfoFile`	The input comma-delimited file containing information on individual stocks. Each row contains ACTIVITY_TS, TICKER, PRICE, VOLUME. See SP500market.csv in the package for an example.

Value

Returns a data frame containing individual dailly stock price info (ACTIVITY_TS, TICKER, PRICE, VOLUME).

Examples

```
## See TestSimulator.R
```

`LoadStocksToDB` *Loads information on individual stocks to R and database*

Description

This function reads in a .csv file containing individual stock's information into the R environment as well as storing the information in a database table.

Usage

```
LoadStocksToDB(schemaDbName, inputStockInfoFile)
```

Arguments

`schemaDbName`	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
`inputStockInfoFile`	The input comma-delimited file containing information on individual daily stock prices. Each row contains TICKER, COMPANY, INDUSTRY. See SP500industry.csv in the package for an example.

Value

Returns a data frame containing individual stock price info (ACTIVITY_TS, TICKER, PRICE, VOLUME).

Examples

```
## See TestSimulator.R
```

`MeanRevertCorrelatePairwise` *Computes correlation information on pair of stocks*

Description

This function computes correlation coefficient for all stock pairs in the stock market dataset and returns the ones that are above the specified correlation thresholds. The results are sorted with highest correlation values at the top. Currently, this function uses the Pearson method on the stock price time-series to compute correlations. This approach has limitations and other techniques may also be used to compute correlations. This is meant to be an illustrative example.

Usage

```
MeanRevertCorrelatePairwise(StockMarketDataset, corrThreshold = 0.95)
```

Arguments

`StockMarketDataset`
: Data frame containing stock market dataset

`corrThreshold`
: Correlation Thresholds for returned result

Value

Returns a data frame of top correlated stock pairs. The dataframe has the following attributes for each row/stock pair: "TICKER1", "TICKER2", "CORR", "ABSCORR", "MEANDIFF", "SDEVDIFF"

Examples

```
## See TestSimulator.R
```

`portfolioBoxPlot` *Plots Stock Prices Box Plot*

Description

Plots Stock Prices Box Plot in our porfolio from low to high prices to identify outliers

Usage

```
portfolioBoxPlot(mData, tData)
```

Arguments

`mData`	Market Data
`tData`	Transaction Data

References

Wickham, Hadley (2009) ggplot2 Chambers, J. (2008) Software for Data Analysis

Examples

```
## see TestVisualizer.R
```

portfolioHoldingSummaryChart *Plots Portfolio Holding Summary Chart*

Description

Compares performance of algorithms used in determining trade strategy in different portfolio holdings

Usage

```
portfolioHoldingSummaryChart(hData, pData, xrng = NULL, yrng = NULL, caption = N
```

Arguments

hData	Historical Stock Market Data
pData	Portfolio Data
xrng,yrng,caption	optional argument of x, y range and caption

References

Wickham, Hadley (2009) ggplot2 Chambers, J. (2008) Software for Data Analysis

Examples

```
## see TestVisualizer.R
```

PortfoliosInfo *Data frame contain information on portfolios*

Description

Data frame containing information on portfolios in simulator

Format

A data frame with the following 3 variables.

PORTFOLIOID Id of portfolio

ALGORITHM name of portfolio

STARTAMOUNT starting investment amount

PERFORMANCE percentage return on investment

Examples

```
## See TestSimulator.R
```

`portfolioStocksChart` *Plots Portfolio Stocks Chart*

Description

Compares performance of different stock prices in our portfolio

Usage

```
portfolioStocksChart(data)
```

Arguments

`data`	Portfolio Data

References

Wickham, Hadley (2009) ggplot2 Chambers, J. (2008) Software for Data Analysis

Examples

```
# see TestVisualizer.R
```

`PrintSimulatorState` *Prints information on the current state of the simulator*

Description

This function displays information such as the time period, nummber of portfolios and other useful general information on the simulator state

Usage

```
PrintSimulatorState()
```

Examples

```
## See TestSimulator.R
```

`readFromDB` *Reads data from sqlite table*

Description

Reads data from sqlite table into a data frame

Usage

```
readFromDB(dbFile, tablename, where = "")
```

Arguments

`dbFile`	Database file path and name
`tablename`	Table name
`where`	SQL where clause

Examples

```
## see TestVisualizer.R
```

`schemaHoldingsTableName` *Database table name for Asset Holdings Info*

Description

Represents the table name for storing Asset Holdings Info

Format

To change this value: schemaHoldingsTableName <- "holdings"

Examples

```
## Setting this value
schemaHoldingsTableName <- "holdings"
```

`schemaMarketTableName`
Database table name for Stock Market Daily Prices Info

Description

Represents the table name for storing Stock Market Daily Prices Info

Format

To change this value: schemaMarketTableName <- "market"

Examples

```
## Setting this value
schemaMarketTableName <- "market"
```

`schemaPortfoliosTableName`
Database table name for Portfolios Info

Description

Represents the table name for storing Portfolios Info

Format

To change this value: schemaPortfoliosTableName <- "portfolios"

Examples

```
## Setting this value
schemaPortfoliosTableName <- "portfolios"
```

`schemaStocksTableName`
Database table name for Individual Stock Info

Description

Represents the table name for storing Individual Stock Info

Format

To change this value: schemaStocksTableName <- "stocks"

Examples

```
## Setting this value
schemaStocksTableName <- "stocks"
```

schemaTransactionsTableName *Database table name for Transaction Info*

Description

Represents the table name for storing Transaction Info

Format

To change this value: schemaTransactionsTableName <- "transactions"

Examples

```
## Setting this value
schemaTransactionsTableName <- "transactions"
```

SellFee *Transaction Fee for Selling a stock*

Description

This represents the value in dollars in transaction fee for selling a stock.

Format

To change the ticker: SellFee <- 1

Examples

```
# To display current SellFee
SellFee
```

SimulateBroker *Simulates the execution of an order book on the stock market*

Description

This function simulates the execution of orders for a Portfolio on the stock market. It outputs dataframes containing updated asset holdings information and executed transactions.

Usage

```
SimulateBroker(portfolioId, PortfoliosInfo, StockMarketDataset, OrderBook, Holdi
```

Arguments

`portfolioId`	Id of Portfolio
`PortfoliosInfo`	Data frame containing information on portfolios in simulator
`StockMarketDataset`	Data frame containing stock market dataset
`OrderBook`	Data frame containing orders generated by trading algorithm
`HoldingsInfo`	Data frame containing information on asset holdings in simulator
`TransactionsInfo`	Data frame containing information on transactions in simulator

Value

Returns a list containing two dataframes: HoldingsInfo and TransactionsInfo

Examples

```
## See TestSimulator.R
```

`stockCorrMatrixChart` *Plots stock correlation matrix chart*

Description

This function initialize and plots stock prices that are highly correlated

Usage

```
stockCorrMatrixChart(data)
```

Arguments

`data`	Correlation matrix data

Examples

```
## see TestVisualizer.R
```

stockCorrPairsCartesianChart
Plots stock correlation cartesian chart

Description

This function initializes and plots stock prices that are highly correlated

Usage

```
stockCorrPairsCartesianChart(sMarketData,corrData)
```

Arguments

sMarketData	Stock Market Data
corrData	Correlation matrix data

Examples

```
## see TestVisualizer.R
```

StoreCorrelationsToDB
Stores Correlations info to database

Description

Stores correlations info in simulator to database

Usage

```
StoreCorrelationsToDB(schemaDbName, dfCorrelationsInfo)
```

Arguments

schemaDbName	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
dfCorrelationsInfo	Data frame in simulator containing portfolios information

StoreHoldingsToDB *Stores asset holdings info to database*

Description

Stores asset holdings info in simulator to database

Usage

```
StoreHoldingsToDB(schemaDbName, dfHoldingsInfo)
```

Arguments

schemaDbName	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
dfHoldingsInfo	Data frame in simulator containing asset holdings information

Examples

```
## See TestSimulator.R
```

StorePortfoliosToDB *Stores portfolios info to database*

Description

Stores portfolios info in simulator to database

Usage

```
StorePortfoliosToDB(schemaDbName, dfPortfoliosInfo)
```

Arguments

schemaDbName	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
dfPortfoliosInfo	Data frame in simulator containing portfolios information

Examples

```
## See TestSimulator.R
```

StoreSimulationResults

Stores simulation results to database

Description

Stores simulation results such PortfoliosInfo, HoldingsInfo, TransactionsInfo to database

Usage

```
StoreSimulationResults(schemaDbName, PortfoliosInfo, HoldingsInfo, TransactionsI
```

Arguments

schemaDbName	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
PortfoliosInfo	Data frame in simulator containing portfolios information
HoldingsInfo	Data frame in simulator containing asset holdings information
TransactionsInfo	Data frame in simulator containing transactions information

Examples

```
## See TestSimulator.R
```

StoreTransactionsToDB

Stores transactions info to database

Description

Stores transactions info in simulator to database

Usage

```
StoreTransactionsToDB(schemaDbName, dfTransactionsInfo)
```

Arguments

schemaDbName	The file path to the .db sqlite database file. The sqlite database should include tables as specified in create_tables.sql included in the package.
dfTransactionsInfo	Data frame in simulator containing transactions information

Examples

```
## See TestSimulator.R
```

TransactionsInfo *Data frame contain information on transactions*

Description

Data frame containing information on transactions in simulator

Format

A data frame with the following 5 variables.

`PORTFOLIOID` Id of Portfolio

`ACTIVITY_TS` timestamp

`DECISION` type of decision: "BUY", "SELL"

`TICKER` stock ticker symbol

`NUMSHARES` number of shares

Examples

```
## See TestSimulator.R
```

VisualizeCorrelation *Visualize the correlation of stock pairs*

Description

This function allows one to plot the stock price time-series of a pair of stocks identified as correlated in the algorithm.

Usage

```
VisualizeCorrelation(pairCorrelationInfo, StockMarketDataset, topPosition)
```

Arguments

`pairCorrelationInfo` Data frame containing pairs of stocks and their pairwise correlation info

`StockMarketDataset` Data frame containing stock market data

`topPosition` The top N-th correlated stock pair to be displayed.

`writeToDb` *Writes data to sqlite table*

Description

Generic function to write data to a sqlite table

Usage

```
writeToDb(dbFile, tablename, df)
```

Arguments

`dbFile`	Database file path and name
`tablename`	Table name
`df`	Data frame

Examples

```
## see TestVisualizer.R
```

Index

∗Topic **datasets**
BuyFee, 3
cashTicker, 5
HoldingsInfo, 10
PortfoliosInfo, 13
schemaHoldingsTableName, 15
schemaMarketTableName, 16
schemaPortfoliosTableName, 16
schemaStocksTableName, 16
schemaTransactionsTableName, 17
SellFee, 17
TransactionsInfo, 22
∗Topic **package**
TradeStrategyAnalyzer-package, 2

AddPortfolio, 3

BuyFee, 3

calcSumHoldings, 4
calendarHeatMap, 4
cashTicker, 5

DetermineOrders_MeanRevert, 5
DetermineOrders_RandomBuyAndHold, 6

GetPortfolioPerformance, 6
GetPortfolioWorthAtTime, 7
gvisAnnotatedTimeLine2, 8
gvisMotionChart2, 9

HoldingsInfo, 10

LoadMarketToDB, 10
LoadStocksToDB, 11

MeanRevertCorrelatePairwise, 11

portfolioBoxPlot, 12
portfolioHoldingSummaryChart, 13
PortfoliosInfo, 13
portfolioStocksChart, 14

PrintSimulatorState, 14

readFromDB, 15

schemaHoldingsTableName, 15
schemaMarketTableName, 16
schemaPortfoliosTableName, 16
schemaStocksTableName, 16
schemaTransactionsTableName, 17
SellFee, 17
SimulateBroker, 17
stockCorrMatrixChart, 18
stockCorrPairsCartesianChart, 19
StoreCorrelationsToDB, 19
StoreHoldingsToDB, 20
StorePortfoliosToDB, 20
StoreSimulationResults, 21
StoreTransactionsToDB, 21

TradeStrategyAnalyzer *(TradeStrategyAnalyzer-package)*, 2
TradeStrategyAnalyzer-package, 2
TransactionsInfo, 22

VisualizeCorrelation, 22

writeToDb, 23