Package 'orderbook'

February 20, 2015

Type Package

Title Orderbook visualization/Charting software

Depends R (>= 2.15.0), methods, graphics, lattice, hash, grid

Version 1.03

Date 2013-04-09

Author Andrew Liu, Khanh Nguyen, David Kane

Maintainer Andrew Liu <andrew.tsutse.liu@gmail.com>

Description Functions for visualizing and retrieving data for the state of an orderbook at a particular period in time.

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Repository/R-Forge/Project limitob

Repository/R-Forge/Revision 227

Date/Publication 2013-04-12 06:40:10

NeedsCompilation no

R topics documented:

`orderbook-package` *Analyze and visualize orderbook data.*

Description

The orderbook package provides functions for exploring and visualizing orderbook data. For example, we may want to study an orderbook immediately before a trade occurs, or maybe 50 orders before a trade occurs, or we want to visualize the change of an orderbook overtime as the price level fluctuates. The insights from this sort of analysis can help in researching short-term trading strategies.

Details

Package:	orderbook
Type:	Package
Version:	1.0
Date:	2010-07-01
Depends:	R (>= 2.11.0), hash, methods, grid, lattice
License:	GPL (>= 2)
LazyLoad:	yes

Index:

Further information is available in the following vignettes:

`orderbook`	Using the orderbook package (source, pdf)

Author(s)

Andrew Liu <andrew.t.liu@williams.edu>, Khanh Nguyen <knguyen@cs.umb.edu>, David Kane <dave@kanecap.com>.

`add.order.function` *add.order*

Description

Modify the order book.

Usage

```
add.order(object, price, size, type, time = NULL, id = NULL, status =
FALSE)
remove.order(object, id)
replace.order(object, id, size)
market.order(object, size, type)
```

Arguments

object	Object of class orderbook
price	of the order to be added.
size	the size of the order to be added. In the case of a replace.order it is the new size of the order. market.order takes this value to be the number of shares to be bought or sold.
type	the user can specify "bid" or "ask." For a market.order the user can specify "buy" or "sell."
time	of the order. If no time is specified the order book will automatically add 1000ms to the current time for the new order.
id	of the order. It must be unique. If no ID is specfied then add.order will automatically add 1 to the largest numeric ID and use it as the new ID.cancel.order and replace.order identify the order to remove/modify using the ID.
status	indicating whether or not the order belongs to the user.

Details

The user can add, remove, and replace orders from the order book. Additionally, the user can issue market orders. These functions are useful for simulating an order book.

Value

Return an object of class orderbook

Examples

```
library(orderbook)
filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")

ob <- orderbook(file = filename)

ob <- add.order(ob, price = 123, size = 123, type = "BID", time = 1, id = 1)
ob <- replace.order(ob, 1, 100)
ob <- market.order(ob, 50, "SELL")
ob <- remove.order(ob, 1)
```

orderbook *Creating an Object of Class Orderbook*

Description

Create an orderbook object from input data.

Usage

```
orderbook(file)
```

Arguments

file	Object of class "character" specifying the location of the data file.

Details

This function should be used to initially create the orderbook function. If you have an orderbook data.frame from a previous session, you can load it and begin from there. If you want to create an empty orderbook, just specify the file (see example).

Value

Return an object of class orderbook

Examples

```
library(orderbook)
filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")
ob <- orderbook(file = filename)
```

orderbook-class *Class "orderbook"*

Description

Functions for manipulating and extractiong information from an orderbook

Objects from the class

Objects can be created by calls to the function orderbook

Slots

`current.ob`: A data frame containing the current state of the orderbook.

`current.time`: A numeric value in milliseconds after midnight that indicates the time of `current.ob`.

`file`: A character string which specifies the location of the file containing the data file.

`file.index`: A numeric value that indicates the current row number of the input file.

`ob.data`: A `hash` that contains all orders currently in the orderbook. Essentially the same as current.ob, but a different data structure.

`trade.data`: A `vector` that contains data on all trades that occurred before current.time.

`trader`: A `logical` indicating whether the order book should expect data with user orders and trades marked.

Methods

show `signature(object = "orderbook")`: Prints the variables used in the orderbook

best.bid `signature(object = "orderbook")`: Returns the current best bid.

best.ask `signature(object = "orderbook")`: Returns the current best ask.

summary `signature(object="orderbook")`: Prints a summary of the orderbook

[`signature(x = "orderbook", i = "character")`: Prints the orders at that price level.

copy `signature(x = "orderbook")`: Creates a copy of the orderbook.

get.order.info `signature(object = "orderbook")`: Returns the price and size of the order with the specified ID.

display `signature(object="orderbook")`: Prints the state of the orderbook. The result is similar to `plot`

bid.price.levels `signature(object="orderbook")`: Returns the number of bid price levels.

ask.price.levels `signature(object="orderbook")`: Returns the number of ask price levels.

total.price.levels `signature(object="orderbook")`: Returns the total number of price levels.

bid.orders `signature(object="orderbook")`: Returns the number of bids.

ask.orders `signature(object="orderbook")`: Returns the number of asks.

total.orders `signature(object="orderbook")`: Returns the total number of orders.

mid.point `signature(object="orderbook")`: Returns the midpoint between the best ask and the best bid.

inside.market `signature(object="orderbook")`: Returns a data frame with a row for the best ask and a row for the best bid. The columns are price, size, and type.

spread `signature(object="orderbook")`: Returns the spread between the best ask and best bid .

reset `signature(object="orderbook")`: Reset the orderbook to its initial state.

Examples

```
library(orderbook)

filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")

ob <- orderbook(file = filename)
ob <- read.orders(ob, 500)

show(ob)
summary(ob)
display(ob)
```

`plot.function` *plot*

Description

Plotting the orderbook.

Usage

```
## S4 method for signature 'orderbook'
plot(x, bounds = 0.1, n = 10, type = "n")
```

Arguments

`x`	Object of class `orderbook`
`bounds`	Percentage above and below the midpoint to determine the y limits.
`n`	Number of price levels to plot.
`type`	Either `"n"`, `"sd"`, `"o"`, or `"s"`.

Details

This function plots the orderbook. `n` is only needed for `type = "s"`, which plots the best bid vs. the best ask, and then the second best bid vs. the second best ask, for `n` bids and asks. If `type = "n"` then the orderbook will be plotted with price levels on the y-axis, and size on the x-axis. `type = "o"` is similar, except the number of orders for each price level are plotted. `type = "sd"` makes a plot that shows the supply and demand curves of the order book.

Value

Prints a Trellis object.

Examples

```
library(orderbook)
filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")

ob <- orderbook(file = filename)
ob <- read.orders(ob, n = 1000)
plot(ob)
plot(ob, type = "o")
plot(ob, type = "sd")
plot(ob, type = "s")
```

`read.orders.function` *read.order*

Description

Read the next n messages of the data file.

Usage

```
read.orders(object, n = 1000)
```

Arguments

`object`	Object of class `orderbook`
`n`	number of messages we want to read.

Details

The orderbook keeps track of its current position within the data file. The read.orders function will read and process the next n messages from the file.

Value

Return an object of class `orderbook`

Examples

```
library(orderbook)
filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")

ob <- orderbook(file = filename)
ob <- read.orders(ob, 100)
```

read.time.function *read.time*

Description

Returns the state of the orderbook at the specified time.

Usage

```
read.time(object, n)
```

Arguments

object	Object of class orderbook
n	A character in the form "HH:MM:SS".

Details

Sets the state of the orderbook to the specified time. Can be used to move backwards or forwards in time.

Value

Return an object of class orderbook

Examples

```
library(orderbook)
filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")

ob <- orderbook(file = filename)
ob <- read.time(ob, "9:30:00")
```

view.trade.function *view.trade*

Description

View the orderbook object when a particular trade happens.

Usage

```
view.trade(object, tradenum)
next.trade(object)
previous.trade(object)
midpoint.return(object, tradenum, time)
```

Arguments

object	Object of class orderbook
tradenum	the trade we want to view.
time	Specifies the number of seconds after the trade to calculate the midpoint return for. Can also be a vector of times.

Details

The orderbook object internally keeps track of the the trades it has read from the data file. The view.trade function returns information about the nth trade. This function can only view trades that occurred prior to the current order.book time.

previous.trade sets the state of the orderbook to the first trade to have happened before the current time. next.trade sets the state to the orderbook at the very next trade to have happened after the current time.

midpoint.return calculates the midpoint return for a trade a specified number of seconds into the future.

Value

Return an object of class orderbook. view.trade and midpoint.return print information to the terminal about the trade and midpoint return, respectively.

Examples

```
library(orderbook)
filename <- system.file("extdata", "sample.txt",
                        package = "orderbook")

ob <- orderbook(file = filename)
ob <- next.trade(ob)
view.trade(ob, 1)
midpoint.return(ob, 1, c(5, 10))
```

Index