# Package 'mobster'

November 4, 2012

**Maintainer** <tom.e.appleton@gmail.com>

**License** GPL-2

**Title** Model Order Book State Explorer (mobster)

**Type** Package

**LazyLoad** true

**Author** Tom Appleton

**Description** A lightweight tool set for modelling exchange orderbook dynamics.
This package maintains an internal order book following the standard Central Limit Order Book algorithm. It currently supports FAK and GTC type orders. Orders can be placed in the book using the 'limit' method and 'market' methods. Quotes could be generated from
historical data, or from a statistical model. Executions in the order book and the full order-book history
can be obtained for analysis purposes. Trades are modelled as a sequence of ordered events. The main engine was designed to be fast for use with large datasets.

**Version** 1.0.0

**Date** 2012-09-25

**Depends** R (>= 2.11.0), data.table (>= 1.7.8), testthat

**Imports** data.table (>= 1.7.8), testthat

**Collate** 'orderbook.R' 'mobster-package.R'

**Archs** i386, x86_64

## R topics documented:

---

`.onLoad` *onload...*

---

## Description

onload

## Usage

```
.onLoad(libname, pkgname)
```

## Arguments

| | |
|---|---|
| `libname` | the library |
| `pkgname` | the pkgname |

---

`cancel` *Cancel an order/quote...*

---

## Description

Cancel an order/quote cancels an order in the orderbook. The order is removed from the book.

## Usage

```
cancel(id, qty=0, time=0, externalId=0)
```

## Arguments

| | |
|---|---|
| `id` | the orderid to cancel. |
| `qty` | the qty of the order to cancel. |
| `time` | the time the order was cancelled (when replaying historical data). |
| `externalId` | the id of the original order (use when replaying historical data). |

## get.execs — *Get the execution history.*

### Description

Get the execution history. Both sides of a trade are recorded in the execution history for analysis. The data.table returned includes the orderid of the order, the matching order id, the qty filled, the price and the current mid price in the book (bid+ask)/2

### Usage

```
get.execs(n)
```

### Arguments

| | |
|---|---|
| n | the number of rows to retrieve |

### Value

data.table with columns: execid, orderid, matchorderid, side, trader, fillqty, price, midp (midprice at time of execution)

## get.filled.qty — *Get the total filled qty for an order.*

### Description

Get the total filled qty for an order.

### Usage

```
get.filled.qty(id, externalId=0)
```

### Arguments

| | |
|---|---|
| id | the orderid to get the filled qty for |
| externalId | the id of the original order (use when replaying historical data). |

## get.hob *Get the historical book.*

### Description

Get the historical book. Each quote/order is recorded in a historical book for analysis. The historical book records 10 levels of bid/ask prices, qty, the total traded volume and total number of trades

### Usage

```
get.hob(n)
```

### Arguments

| | |
|---|---|
| n | the number of orders to retrieve |

## get.ob *Get the orderbook.*

### Description

Get the orderbook. Gets a snapshot of the current orderbook. The order book has 10 ask and 10 bid levels. By adding limit orders/quotes and market orders the dynamics of an exhange can be modelled. By using a file of real exchange data trading strategies can be plugged in to model market microstructure.

## init.book *reset the book...*

### Description

reset the book clears all the values in the historical order book, clears the execution history, and resets the current orderbook. This should be called everytime before starting a new session. The book is started at 100.00/99.99 by default.

### Usage

```
init.book(n=10000, ask=100, bid=99.99)
```

### Arguments

| | |
|---|---|
| n | int. The number of rows to initialise in the book |
| ask | The starting ask price |
| bid | The starting bid price |

## limit — *Add limit order...*

### Description

Add limit order adds a limit order to the book. The order will match if it crosses. Types include FAK (fill and kill) or GTC (good till cancel). GTC types will remain in the orderbook until filled or cancelled.

### Usage

```
limit(sym, trader, side, price, qty, tradetype, time=0, externalId=0)
```

### Arguments

| | |
|---|---|
| sym | string. The symbol of the instrument being traded |
| trader | string. The trader (or strategy name) making the trade |
| side | int. The side of the trade. buy=0, sell=1 |
| qty | int. The qty to be traded |
| price | double. The limit price |
| tradetype | string. GTC or FAK |
| time | int. The time a trade was done (ms) |
| externalId | int. The trade id. Defaulted to zero, but passed so that historical replay from total itch data can be used to cancel orders correctly. |

## market — *Place a market order.*

### Description

Place a market order. A market order will scan the orderbook until the amount required has been matched, or there is no more depth in the orderbook.

### Usage

```
market(sym, trader, side, price, qty, time=0, externalId=0)
```

### Arguments

| | |
|---|---|
| sym | string. The symbol of the instrument being traded |
| trader | string. The trader (or strategy name) making the trade |
| side | int. The side of the trade. buy=0, sell=1 |
| price | double. not used |
| qty | int. The qty to be traded |
| time | the time the order was cancelled (when replaying historical data). |
| externalId | the id of the original order (use when replaying historical data). |

mobster *Model Order Book State Explorer.*

## Description

Model Order Book State Explorer.

## Details

A lightweight tool set for modelling exchange orderbook dynamics. This package maintains an internal order book following the standard Central Limit Order Book algorithm. It currently supports FAK and GTC type orders.

Orders can be placed in the book using the 'limit' method and 'market' methods. Quotes could be generated from historical data, or from a statistical model. Executions in the order book and the full orderbook history can be obtained for analysis purposes. Trades are modelled as a sequence of ordered events.

The main engine was designed to be fast for use with largish datasets (up to 1000000 records currently).

Uses include modelling market impact of trades, predicting market moves based on orderbook microstructure, analysing trading strategies against an orderbook, modelling trade arrival etc.

Currently only one symbol is supported (ie one instrument). Future versions may be enhanced to model multiple symbols.

## Examples

```
# first initialise the book (sets the start price, and how many historical rows to clear)
init.book(1000)

# get the orderbook
ob <- get.ob()

# add some sell limit orders - fill multiple levels
limit("sym", "trader", 1, 100.00, 1, "GTC")
limit("sym", "trader", 1, 100.01, 1, "GTC")
limit("sym", "trader", 1, 100.02, 1, "GTC")
limit("sym", "trader", 1, 100.03, 1, "GTC")
limit("sym", "trader", 1, 100.04, 1, "GTC")

ob <- get.ob()

# get the historical order book
hob <- get.hob(10)

# cross (ie execute a trade)
id <- limit("sym", "trader", 0, 100.02, 1, "GTC")

# crossed at the lowest price level - ie 100.00. Check the execution history.
# both sides are reoprted
get.execs(10)

# get the filled qty for the order
get.filled.qty(id)
```

```
# see applett/rstuff for examples of how the toolkit
# can be used to perform more sophisticated scenarios.
```

# Index