Package 'MOBSTER'

September 24, 2012

Maintainer Who to complain to <tom.e.appleton@gmail.com>

License

Title Model Order Book State Explorer (MOBSTER)

Type Package

LazyLoad true

Author Tom Appleton

Description A package for modelling exchange orderbook dynamics.

Version 1.0

Date 2012-08-04

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

Imports data.table (>= 1.7.8)

Collate 'depth-strategy.R' 'market-making-strategy.R''mavg-strategy.R' 'orderbook.R' 'poisson-book.R''MOBSTER-package.R'

Archs i386, x86_64

R topics documented:

`.onLoad` *.onLoad*

Usage

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

Arguments

`id`	the orderid to cancel.

`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)
```

Arguments

id	the orderid to get the filled qty for

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(rsym, rtrader, rside, rprice, rqty, rtradetype)
```

Arguments

rsym	string. The symbol of the instrument being traded
rtrader	string. The trader (or strategy name) making the trade
rside	int. The side of the trade. buy=0, sell=1
rqty	int. The qty to be traded
rprice	double. The limit price
rtradetype	string. GTC or FAK

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)
```

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

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, and the notion of 'time' between trade arrival events is currently not considered.

The main engine was designed to be fast for use with large datasets.

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)

# the following examples show how the base
# toolkit can be used to perform more sophisticated scenarios.

# models trade arrival from poisson distribution
?model.poisson
model.poisson(100)
plot(get.hob(100)[2:1000]$ask0, type='s')

# a bid/offer market making strategy that provides liquidity
?trade.mm
trade.mm(100)

# now analyse the performance
?trade.mm.analyse
trade.mm.analyse(100)
```

`model.poisson` *Model the book as Poisson events.*

Description

Model the book as Poisson events. This demonstrates constructing a book from limit orders and trades generated by a poisson process.

Usage

```
model.poisson(num=100)
```

Arguments

num	The number of iterations to model

Examples

```
model.poisson(100)
plot(get.hob(100)[2:100]$ask0, type='s')
```

---

trade.depth *Trades when depth is thin in one side of the market.*

Description

Trades when depth is thin in one side of the market. Generates quote events according to poisson, and then analyses the book to determine when to trade. Trades are taken when the depth on one side of the book is larger than the other, and hence more likely to move. This strategy can be elaborated on to look at futures prices vs spot or by using a bayesian updating probability of when the market will move considering multiple quote levels

Usage

```
trade.depth(num=100)
```

Arguments

num	The number of iterations to model

---

trade.mavg *simple mavg trading for comparison (ie to show how terribly it performs!)...*

Description

simple mavg trading for comparison (ie to show how terribly it performs!) generate quote events according to poisson, and then use moving avg to enter/exit positions

Usage

```
trade.mavg(num=100)
```

Arguments

num	The number of iterations to model

`trade.mavg.sumpnl` *sum the profit and loss...*

Description

sum the profit and loss perform some analysis on the moving avg strategy. It uses the trader id for the different scenarios traded (cs='close sell',cb='close buy',os='open sell',ob='open buy') and calculates where the trade was entered and exited.

Usage

```
trade.mavg.sumpnl(x)
```

Arguments

`x`	the executions we made

`trade.mm` *Simple market making strategy.*

Description

Simple market making strategy. Whenever the spread is wider than a target spread this strategy provides liquidity. This strategy attempts to keep risk (the position size) close to zero.

Usage

```
trade.mm(num=100)
```

Arguments

`num`	The number of iterations to model

`trade.mm.analyse` *analyse the market making strategy trades.*

Description

analyse the market making strategy trades. this function gets all the executions and extracts the deals done by the market making function. The pnl analysis calculates the spread pnl (how much is earned by providing liquidity) and the costs of inventory holding.

Usage

```
trade.mm.analyse(numobs)
```

Arguments

`numobs`	The number of observations to analyse

Index