

orderbook

Andrew Liu, Khanh Nguyen and David Kane

Introduction

The **orderbook** package provides functions for exploring and visualizing orderbook data. Using the **orderbook** package, analysts can create a object of class `orderbook`, that is able to accept add, replace, cancel, and market orders. From this object summary statistics such as the spread, number of price levels, and depth of the market can be retrieved.

Background

An orderbook keeps track of the outstanding limit orders for a current stock. For example, person A may want to purchase 100 shares of IBM at \$11.00, in which case A would submit a limit order. In the orderbook for IBM this would be represented as a *bid* at \$11.00 for 100 shares. Suppose person B submits the same order. Then there would be 200 shares offered at \$11.00. Next, consider a Person C wants to sell 150 shares of IBM at \$11.05. The orderbook now has an *ask* order.

The best ask/bid is the lowest/highest price level that individuals are willing to sell/buy stocks. In this case the best ask is \$11.05, and the best bid is \$11.00, and the *spread*, which is the difference between the two, is \$0.05. The *depth*, or number of shares being offered at each price level, is 150 at \$11.05, and 200 at \$11.00. We call this the *inside market*. Another noteworthy statistic is the *midpoint*, which is the simple average of the best bid and best ask.

Now if A decided he only wanted to buy 50 shares of IBM he could submit a replace order to reduce the number of shares he is willing to purchase. Note that a replace order can never increase the number of shares to be purchased or sold. The reason for this is that in the event of a market order, the orders submitted earliest are processed (filled) first. Now perhaps B decides that he wants to buy IBM at \$10.95 instead of \$11.00. B would cancel his order, then submit a new limit order at the lower price. This would result in two *price levels* on the bid side.

A person D may come in and submit a market order to sell 125 shares of IBM. With a market order, price does not matter, and transactions are carried out until the desired number of shares to be bought or sold is reached. In this case, A will buy 50 shares at \$11.00, and B will buy 75 shares at \$10.95.

Hundreds of thousands of orders are submitted daily for a single stock, and the orderbook package allows the user to examine the orderbook at any given point during the day.

Examples

Consider the first 10,000 orders for a NVDA on June 8, 2010. Start by loading and examining the input data.

```
> library(orderbook)

> ob = orderbook(file = "sample.txt")
> ob = read.orders(ob, 10000)
> ob
```

An object of class `orderbook`

```
-----
Current orderbook time:    09:35:02
File Index:                10000
Number of Bids:            631
Number of Asks:            1,856
Total Orders:              2,487
```

We create the orderbook object by giving the object the location of our data file. Then we read in the first 10,000 orders. Only 2,487 orders remain, indicating that many have either been cancelled or removed through trades.

```
> summary(ob)
```

Current time is 09:35:02

```
ASK price levels:    540
BID price levels:    179
Total price levels:  719
```

```
-----
ASK orders:          1856
BID orders:          631
Total orders:        2487
```

```
-----
Spread:              0.02
```

```
Mid point:           11.370
```

Inside market

```
Best Bid:            11.36
Size:                 2,700.00
```

```
Best Ask:            11.38
Size:                 400.00
```

In addition to the number of orders, with summary we also see the number of price levels, spread, midpoint, and inside market.

```
> display(ob)
```

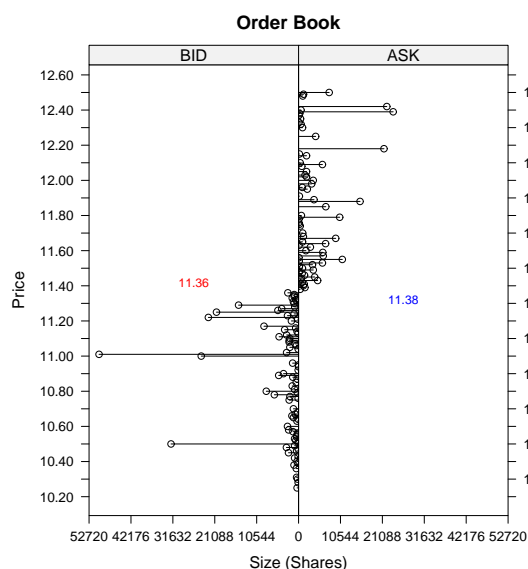
Current time is 09:35:02

Price	Ask Size
11.42	900
11.41	1,400
11.40	1,205
11.39	1,600

	11.38	400
<hr/>		
2,700	11.36	
1,100	11.35	
1,100	11.34	
1,600	11.33	
700	11.32	
<hr/>		
Bid Size	Price	

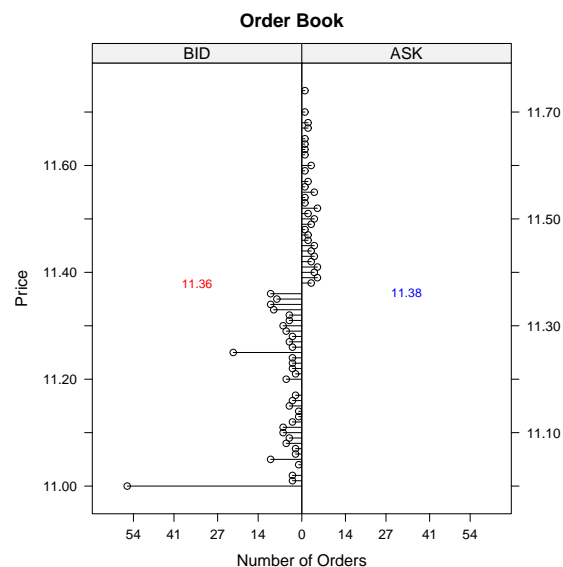
`summary` provides the user with a set of important summary statistics, while `display` shows the top five bid and ask price levels, along with the corresponding depth. Finally, `plot` shows a graphical representation of the orderbook with price levels on the y-axis, and size on the x-axis. By default, the maximum and minimum price levels are 10% above and below the midpoint price, respectively.

```
> plot(ob)
```



Note that there is a particularly large order at \$11.01. It is helpful to know whether the depth at that price level is comprised of a single order, or several. Additionally, we lower the bounds since we are really only concerned with the \$11.01 price level.

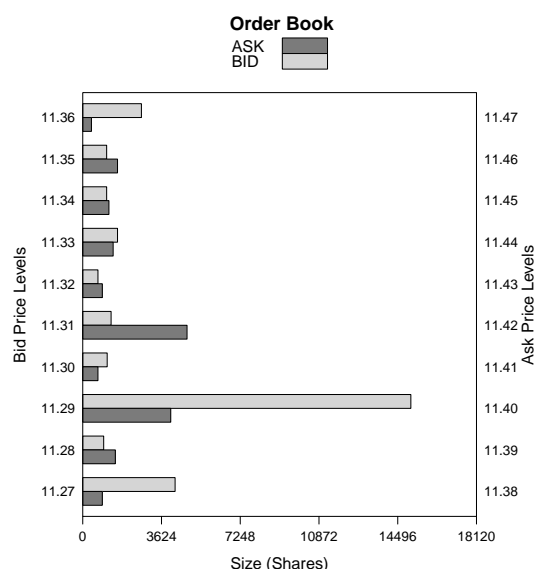
```
> plot(ob, bounds = 0.033, type = "o")
```



At first it appears that there are many orders at \$11.01, but upon closer examination we see that there are 56 orders at \$11.00, but only 3 at \$11.01, indicating that the depth at that price level is composed of a few very large orders (in this case one order for 50,000).

Viewing the orderbook with bids on one side and asks on another is useful, but sometimes users may want to view them side by side to more directly compare the supply and demand at each price level.

```
> plot(ob, type = "s")
```



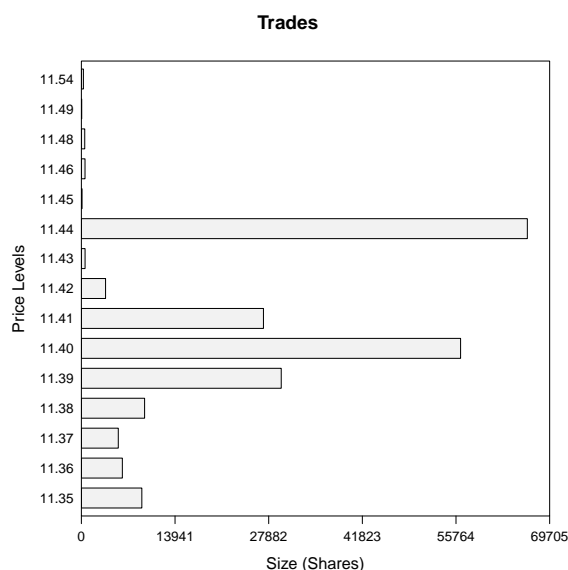
Additionally, The user can view a simple animation of the orderbook between two times.

```
> animate.ob(ob, "9:30:00", "9:32:00")
```

Finally, users can easily plot the trade data by using `plot.trade`. This creates a simple bar graph of

the number of shares traded at each price level.

```
> plotTrade(ob)
```



Aside from the ability to easily retrieve summary statistics and create graphics, **orderbook** can create different orderbook objects for viewing the orderbook at different times. For example, the user may want to view the orderbook when the market opens.

```
> ob = read.time(ob, "9:30:00")
```

Suppose the user wants to view the last pre-market trade. `previous.trade` returns the state of the orderbook at the time of the trade immediately preceding the current time occurred, and `next.trade` returns the state of the orderbook at the time of the next trade.

```
> next.ob = next.trade(ob)
> prev.ob = prev.trade(ob)
```

To look at the 50 orders preceding `next.ob` or following `next.ob`, we use the `read.orders` command.

```
> a = read.orders(next.ob, n = 50)
> b = read.orders(next.ob, n = -50)
```

Data

Almost every broker and exchange has its own way of transmitting raw order data to customers, so it would be unfeasible for us to write scripts to automatically process that data. Consequently, raw data for an orderbook object should be in the following form:

```
A,31285893,1231884,11.49,200,ASK
R,31295779,1231884,150
T,31295779,1231884,11.49,50,BUY
C,31295781,1231884
```

where A, R, T, C mean Add, Replace, Trade, and Cancel order, respectively. The first number is the timestamp of the order in milliseconds after midnight of the users timezone, and the second number (or string) is the ID of the order. For a Replace the next number is the new size, while for Add and Trade price comes before size, followed by the type of order/transaction (BID/ASK or BUY/SELL).

Simulation

The orderbook object supports adding, replacing, and cancelling orders. To add an order, the user needs to specify the price, size, and type. Time and ID are optional, and will default to the maximum time and the maximum ID + 1, respectively. For replacing an order, only ID and size need to be given, and for cancelling an order, only ID is necessary. Market orders are also possible by specifying the size and side (BUY/SELL).

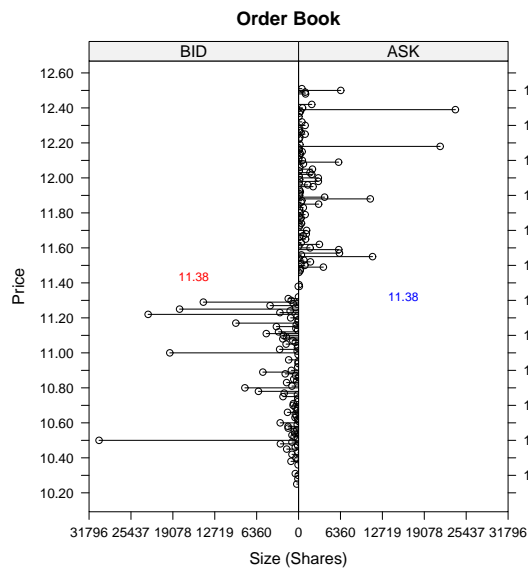
```
> display(ob)
> ob = add.order(ob, stuff)
> ob = remove.order(ob, stuff)
> ob = replace.order(ob, stuff)
> ob = market.order(ob, 200, "BUY")
> display(ob)
```

Using these tools, the user can write functions to simulate the movement of an orderbook.

In the following example, we consulted Gilles (2006). We simulate 1000 orders. In each iteration there is a 50% chance for a cancel order to be placed, 20% chance for a market order, and 30% chance for a limit order. Orders are cancelled completely randomly, and for a market order there is a 50-50 chance for a buy or sell order to be placed. The size of the market order always corresponds to the size of the best ask or bid at the front of the queue. When a limit order is placed, there is a 50-50 chance for it to be an ask or bid. Then there is a 35% chance for the price to be within the spread, in which case a price is chosen based on a uniform distribution. If the price is determined to be outside of the spread, a price is chosen using a power law distribution. The size follows a log-normal distribution.

```
> ob = simulate(ob)
```

```
> plot(ob)
```



Conclusion

In this article, we have described the orderbook package. orderbook aims to provide user-friendly statistical and visualization tools for analyzing orderbooks. We demonstrated the functionality of the package through a series of examples. Users who deal frequently with orderbook data (i.e high frequency traders) will hopefully find the package useful. While the current package is not complex enough to be a stand-alone platform for developing trading strategies, it can be useful for generating ideas for strategies.

Andrew Liu, Khanh Nguyen and David Kane

Kane Capital Management

Cambridge, Massachusetts, USA

Andrew.T.Liu@williams.edu, knguyen@cs.umb.edu,
and dave@kanecap.com

Bibliography

D. Gilles. *Asynchronous Simulations of a Limit Order Book*. PhD thesis, University of Manchester, 2006.