Contents

Overview 1
   Recommended environment settings ........................................... 1

Loading data 2
   Expected csv schema .............................................................. 2
   Preprocessed example data ..................................................... 2

Visualisation 6
   Order book shape ................................................................. 7
   Price level volume ............................................................... 7
   Liquidity ............................................................................. 13
   Order cancellations ............................................................. 15

Analysis 18
   Order book reconstruction ....................................................... 18
   Market impacts ....................................................................... 18

Overview

*obAnalytics* is an R package intended for visualisation and analysis of limit order data.

This guide is structured as an end-to-end walk-through and is intended to demonstrate the main features and functionality of the package.

Recommended environment settings

Due to the large number of columns in the example data, it is recommended to set the display width to make the most use of the display. It is also recommended to set `digitssecs=3` and `scipen=999` in order to display timestamps and fractions nicely. This can be achieved as follows:

```r
max.cols <- Sys.getenv("COLUMN")
options(width=if(max.cols !="") max.cols else 80, scipen=999, digitssecs=3)
```
Loading data

The main focus of this package is reconstruction of a limit order book. The `processData` function will perform data processing based on a supplied CSV file, the format of which is defined in the *Expected csv schema* section.

The data processing consists of a number of stages:

1. Cleaning of duplicate and erroneous data.
2. Identification of sequential event relationships.
3. Inference of trade events via order-matching.
4. Inference of order types (limit vs market).
5. Construction of volume by price level series.

Limit order events are related to one another by volume deltas (the change in volume for a limit order). To simulate a matching-engine, and thus determine directional trade data, volume deltas from both sides of the limit order book are ordered by time, yielding a sequence alignment problem, to which the Needleman-Wunsch algorithm has been applied.

```
# load and process example csv data from the package inst/extdata directory.
csv.file <- system.file("extdata", "orders.csv.xz", package="obAnalytics")
lob.data <- processData(csv.file)
```

**Expected csv schema**

The CSV file is expected to contain 7 columns:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Numeric limit order unique identifier.</td>
</tr>
<tr>
<td>timestamp</td>
<td>Time in milliseconds when event received locally.</td>
</tr>
<tr>
<td>exchange.timestamp</td>
<td>Time in milliseconds when order first created on the exchange.</td>
</tr>
<tr>
<td>price</td>
<td>Price level of order event.</td>
</tr>
<tr>
<td>volume</td>
<td>Remaining order volume.</td>
</tr>
<tr>
<td>action</td>
<td>Event action describes the limit order lifecycle. One of: created, modified, deleted.</td>
</tr>
</tbody>
</table>

**Preprocessed example data**

For illustrative purposes, the package contains a sample of preprocessed data. The data, taken from the Bitstamp (bitcoin) exchange on 2015-05-01, consists of 50,393 limit order events and 482 trades occurring from midnight up until ~5am.

The sample data, which has been previously processed by the `processData` function, may be attached to the environment with the `data()` function:
The `lob.data` object is a list containing four data.frames.

Table 2: lob.data summary.

<table>
<thead>
<tr>
<th>data.frame</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>events</strong></td>
<td>Limit order events.</td>
</tr>
<tr>
<td><strong>trades</strong></td>
<td>Inferred trades (executions).</td>
</tr>
<tr>
<td><strong>depth</strong></td>
<td>Order book price level depth through time.</td>
</tr>
<tr>
<td><strong>depth.summary</strong></td>
<td>Limit order book summary statistics.</td>
</tr>
</tbody>
</table>

The contents of which are briefly discussed in the following sections.

**Events**

The `events` data.frame contains the lifecycle of limit orders and makes up the core data of the obAnalytics package. Each row corresponds to a single limit order action, of which three types are possible.

Table 3: Possible limit order actions.

<table>
<thead>
<tr>
<th>Event action</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>created</td>
<td>The order is created with a specified amount of volume and a limit price.</td>
</tr>
<tr>
<td>changed</td>
<td>The order has been partially filled. On each modification, the remaining volume will decrease.</td>
</tr>
<tr>
<td>deleted</td>
<td>The order may be deleted at the request of the trader or, in the event that the order has been completely filled, deleted by the exchange. An order deleted by the exchange as a result of being filled will have 0 remaining volume at time of deletion.</td>
</tr>
</tbody>
</table>

In addition to the event `action` type, a row consists of a number of attributes relating to the lifecycle of a limit order.

Table 4: Limit order event attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>event.id</td>
<td>Event Id.</td>
</tr>
<tr>
<td>id</td>
<td>Limit Order Id.</td>
</tr>
<tr>
<td>timestamp</td>
<td>Local event timestamp (local time the event was observed).</td>
</tr>
<tr>
<td>exchange.timestamp</td>
<td>Exchange order creation time.</td>
</tr>
<tr>
<td>price</td>
<td>Limit order price level.</td>
</tr>
<tr>
<td>volume</td>
<td>Remaining limit order volume.</td>
</tr>
<tr>
<td>action</td>
<td>Event action: <code>created</code>, <code>changed</code>, <code>deleted</code>. (as described above).</td>
</tr>
<tr>
<td>fill</td>
<td>For <code>changed</code> or <code>deleted</code> events, indicates the change in volume between this event and the last.</td>
</tr>
<tr>
<td>matching.event</td>
<td>Matching <code>event.id</code> if this event is part of a trade. NA otherwise.</td>
</tr>
<tr>
<td>type</td>
<td>Limit order type (see Event types below.)</td>
</tr>
</tbody>
</table>
**Attribute Meaning**

**aggressiveness.bps** The distance of the order from the edge of the book in Basis Points (BPS). If an order is placed exactly at the best bid/ask queue, this value will be 0. If placed behind the best bid/ask, the value will be negative. A positive value is indicative of an innovative order: The order was placed inside the bid/ask spread, which would result in the change to the market midprice.

An individual limit order (referenced by the id attribute) may be of six different types, all of which have been classified by onAnalytics.

<table>
<thead>
<tr>
<th>Limit order type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>unknown</td>
<td>It was not possible to infer the order type given the available data.</td>
</tr>
<tr>
<td>flashed-limit</td>
<td>Order was created then subsequently deleted. 96% of example data. These types of orders are also referred to as fleeting orders in the literature.</td>
</tr>
<tr>
<td>resting-limit</td>
<td>Order was created and left in order book indefinitely until filled.</td>
</tr>
<tr>
<td>market-limit</td>
<td>Order was partially filled before landing in the order book at it’s limit price. This may happen when the limit order crosses the book because, in the case of a bid order, it’s price is &gt;= the current best ask. However there is not enough volume between the current best ask and the order limit price to fill the order’s volume completely.</td>
</tr>
<tr>
<td>market</td>
<td>Order was completely filled and did not come to rest in the order book. Similarly to a market-limit, the market order crosses the order book. However, it’s volume is filled before reaching it’s limit price. Both market-limit and market orders are referred to as marketable limit orders in the literature.</td>
</tr>
<tr>
<td>pacman</td>
<td>A limit-price modified in situ (exchange algorithmic order). The example data contains a number of these order types. They occur when a limit order’s price attribute is updated. In the example data, this occurs from a special order type offered by the exchange which, in the case of a bid, will peg the limit price to the best ask once per second until the order has been filled.</td>
</tr>
</tbody>
</table>

The following table demonstrates a small snapshot (1 second) of event data. Some of the attributes have been omitted or renamed for readability.

```r
one.sec <- with(lob.data, {
  events[events$timestamp >= as.POSIXct("2015-05-01 04:55:10", tz="UTC") &
           events$timestamp <= as.POSIXct("2015-05-01 04:55:11", tz="UTC"), ]
})
one.sec$volume <- one.sec$volume*10^-8
one.sec$fill <- one.sec$fill*10^-8
one.sec$aggressiveness.bps <- round(one.sec$aggressiveness.bps, 2)
one.sec <- one.sec[, c("event.id", "id", "price", "volume", "action",
                        "direction", "fill", "matching.event", "type", "aggressiveness.bps")]
colnames(one.sec) <- c("event.id", "id", "price", "vol", "action", "dir",
                        "fill", "match", "type", "agg")
print(one.sec, row.names=F)
```
The package automatically infers execution/trade events from the provided limit order data.

In addition to the usual timestamp, price and volume information, each row also contains the trade direction (buyer or seller initiated) and maker/taker limit order ids.

The maker/taker event and limit order ids can be used to group trades into market impacts - An example of which will be demonstrated later in this guide.

```
trades.ex <- tail(lob.data$trades, 10)
trades.ex$volume <- round(trades.ex$volume*10^-8, 2)
print(trades.ex, row.names=F)
```

<table>
<thead>
<tr>
<th>timestamp</th>
<th>price</th>
<th>volume</th>
<th>direction</th>
<th>maker.event.id</th>
<th>taker.event.id</th>
<th>maker</th>
<th>taker</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-05-01 04:59:27.503</td>
<td>335.73</td>
<td>0.01</td>
<td>buy</td>
<td>49630</td>
<td>49777</td>
<td>65619731</td>
<td>65619806</td>
</tr>
<tr>
<td>2015-05-01 04:59:27.532</td>
<td>335.79</td>
<td>0.02</td>
<td>buy</td>
<td>49672</td>
<td>49778</td>
<td>65619752</td>
<td>65619806</td>
</tr>
<tr>
<td>2015-05-01 04:59:41.568</td>
<td>335.77</td>
<td>0.02</td>
<td>buy</td>
<td>49807</td>
<td>49821</td>
<td>65619818</td>
<td>65619826</td>
</tr>
<tr>
<td>2015-05-01 04:59:55.877</td>
<td>335.77</td>
<td>0.02</td>
<td>buy</td>
<td>49803</td>
<td>49871</td>
<td>65619818</td>
<td>65619851</td>
</tr>
<tr>
<td>2015-05-01 04:59:59.217</td>
<td>335.77</td>
<td>0.38</td>
<td>buy</td>
<td>49804</td>
<td>49877</td>
<td>65619818</td>
<td>65619854</td>
</tr>
<tr>
<td>2015-05-01 05:00:08.361</td>
<td>335.77</td>
<td>0.12</td>
<td>sell</td>
<td>49878</td>
<td>49894</td>
<td>65619854</td>
<td>65619862</td>
</tr>
<tr>
<td>2015-05-01 05:00:08.395</td>
<td>335.58</td>
<td>0.21</td>
<td>sell</td>
<td>49406</td>
<td>49895</td>
<td>65619615</td>
<td>65619862</td>
</tr>
<tr>
<td>2015-05-01 05:00:08.424</td>
<td>335.01</td>
<td>0.07</td>
<td>sell</td>
<td>49221</td>
<td>49896</td>
<td>65618028</td>
<td>65619862</td>
</tr>
<tr>
<td>2015-05-01 05:00:10.108</td>
<td>335.79</td>
<td>0.02</td>
<td>buy</td>
<td>49816</td>
<td>49900</td>
<td>65619824</td>
<td>65619864</td>
</tr>
<tr>
<td>2015-05-01 05:03:13.566</td>
<td>335.45</td>
<td>0.05</td>
<td>sell</td>
<td>49992</td>
<td>50255</td>
<td>65619912</td>
<td>65620048</td>
</tr>
</tbody>
</table>

Each row, representing a single trade, consists of the following attributes:

Table 8: Trade data attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| timestamp | Local event timestamp.
Attribute | Meaning
---|---
price | Price at which the trade occurred.
volume | Amount of traded volume.
direction | The trade direction: *buy* or *sell*.
maker.event.id | Corresponding market making event id in *events* data.frame.
taker.event.id | Corresponding market taking event id in *events* data.frame.
maker | Id of the market making limit order in *events* data.frame.
taker | Id of the market taking limit order in *events* data.frame.

Depth

The depth data.frame describes the amount of available volume for all price levels in the limit order book through time. Each row corresponds to a limit order event, in which volume has been added or removed. The data.frame represents a run-length-encoding of the cumulative sum of depth for all price levels and consists of the following attributes:

Table 9: Depth attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>Time at which volume was added or removed.</td>
</tr>
<tr>
<td>price</td>
<td>Order book price level.</td>
</tr>
<tr>
<td>volume</td>
<td>Amount of remaining volume at this price level.</td>
</tr>
<tr>
<td>side</td>
<td>The side of the price level: bid or ask.</td>
</tr>
</tbody>
</table>

Depth summary

The depth.summary data.frame contains various summary statistics describing the state of the order book after every limit order event. The metrics are intended to quantify the shape of the order book through time.

Table 10: Order book summary metrics.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>Local timestamp corresponding to events.</td>
</tr>
<tr>
<td>best.bid.price</td>
<td>Best bid price.</td>
</tr>
<tr>
<td>best.bid.vol</td>
<td>Amount of volume available at the best bid.</td>
</tr>
<tr>
<td>bid.vol25:500bps</td>
<td>The amount of volume available for 20 25bps percentiles below the best bid.</td>
</tr>
<tr>
<td>best.ask.price</td>
<td>The best ask price.</td>
</tr>
<tr>
<td>best.ask.vol</td>
<td>Amount of volume available at the best ask.</td>
</tr>
<tr>
<td>ask.vol25:500bps</td>
<td>The amount of volume available for 20 25bps percentiles above the best ask.</td>
</tr>
</tbody>
</table>

Visualisation

The package provides a number of functions for the visualisation of limit order events and order book liquidity. The visualisations all make use of the ggplot2 plotting system.
Order book shape

The purpose of the cumulative volume graph is to quickly identify the shape of the limit order book for the given point in time. The “shape” is defined as the cumulative volume available at each price level, starting at the best bid/ask.

Using this shape, it is possible to visually summarise order book imbalance and market depth.

```r
# get a limit order book for a specific point in time, limited to +- 150bps
# above/below best bid/ask price.
lob <- orderBook(lob.data$events,
    tp=as.POSIXct("2015-05-01 04:38:17.429", tz="UTC"), bps.range=150)

# visualise the order book liquidity.
plotCurrentDepth(lob, volume.scale=10^-8)
```

In the figure above, an order book has been reconstructed with the `orderBook` function for a specific point in time. The visualisation produced with the `plotCurrentDepth` function depicts a number of order book features. Firstly, the embedded bar chart at the bottom of the plot shows the amount of volume available at specific price levels ranging from the bid side on the left (blue) through to the ask side (red) on the right. Secondly, the blue and red lines show the cumulative volume of the bar chart for the bid and ask sides of the order book respectively. Finally, the two subtle vertical lines at price points $234 and $238 show the position of the top 1% largest limit orders.

Price level volume

The available volume at each price level is colour coded according to the range of volume at all price levels. The colour coding follows the visible spectrum, such that larger amounts of volume appear “hotter” than smaller amounts, where cold = blue, hot = red.
Since the distribution of limit order size exponentially decays, it can be difficult to visually differentiate: most values will appear to be blue. The function provides price, volume and a colour bias range to overcome this.

Setting \texttt{col.bias} to 0 will colour code volume on the logarithmic scale, while setting \texttt{col.bias < 1} will “squash” the spectrum. For example, a uniform \texttt{col.bias} of 1 will result in 1/3 blue, 1/3 green, and 1/3 red applied across all volume - most values will be blue. Setting the \texttt{col.bias} to 0.5 will result in 1/7 blue, 2/7 green, 4/7 red being applied such that there is greater differentiation amongst volume at smaller scales.

```r
# plot all lob.data price level volume between $233 and $245 and overlay the # market midprice.
spread <- getSpread(lob.data$depth.summary)
plotPriceLevels(lob.data$depth, spread, price.from=233, price.to=245, volume.scale=10^-8, col.bias=0.25, show.mp=T)
```

The above plot shows all price levels between $227 and $245 for ~5 hours of price level data with the market midprice indicated in white. The volume has been scaled down from Satoshi to Bitcoin for legibility (1 Bitcoin = 10^8 Satoshi). Note the large sell/ask orders at $238 and $239 respectively.

Zooming into the same price level data, between 1am and 2am and providing \textit{trades} data to the plot will show trade executions. In the below plot which has been centred around the bid/ask spread, shows the points at which market \textit{sell} (circular red) and \textit{buy} (circular green) orders have been executed with respect to the order book price levels.

```r
# plot 1 hour of trades centred around the bid/ask spread.
plotPriceLevels(lob.data$depth, trades=lob.data$trades, price.from=236, price.to=237.75, volume.scale=10^-8, col.bias=0.2, start.time=as.POSIXct("2015-05-01 01:00:00.000", tz="UTC"), end.time=as.POSIXct("2015-05-01 02:00:00.000", tz="UTC"))
```
Zooming in further to a 30 minute window, it is possible to display the bid ask spread clearly. In the below plot, `show.mp` has been set to FALSE. This has the effect of displaying the actual spread (bid = green, ask = red) instead of the (bid+ask)/2 midprice.

```r
# zoom in to 30 minutes of bid/ask quotes.
plotPriceLevels(lob.data$depth, spread, price.from=235.25, price.to=237,
    start.time=as.POSIXct("2015-05-01 00:45:00.000", tz="UTC"),
    end.time=as.POSIXct("2015-05-01 01:15:00.000", tz="UTC"),
    volume.scale=10^-8, col.bias=0.5, show.mp=F)
```
Zooming in, still further, to ~4 minutes of data focussed around the spread, shows (in this example) the bid rising and then dropping, while, by comparison, the ask price remains static.

This is a common pattern: 2 or more algorithms are competing to be ahead of each other. They both wish to be at the best bid+1, resulting in a cyclic game of leapfrog until one of the competing algorithm withdraws, in which case the remaining algorithm “snaps” back to the next best bid (+1). This behavior results in a sawtooth pattern which has been characterised by some as market- manipulation. It is simply an emergent result of event driven limit order markets.

```r
# zoom in to 4 minutes of bid/ask quotes.
plotPriceLevels(lob.data$depth, spread, price.from=235.90, price.to=236.25,
                start.time=as.POSIXct("2015-05-01 00:55:00.000", tz="UTC"),
                end.time=as.POSIXct("2015-05-01 00:59:00.000", tz="UTC"),
                volume.scale=10^-8, col.bias=0.5, show.mp=F)
```
By filtering the price or volume range it is possible to observe the behavior of individual market participants. This is perhaps one of the most useful and interesting features of this tool.

In the below plot, the displayed price level volume has been restricted between 8.59 an 8.72 bitcoin, resulting in obvious display of an individual algo. Here, the algo. is most likely seeking *value* by placing limit orders below the bid price waiting for a market impact in the hope of reversion by means of market resilience (the rate at which market makers fill a void after a market impact).

```r
plotPriceLevels(lob.data$depth, spread, price.from=232.5, price.to=237.5,
                 volume.scale=10^-8, col.bias=1, show.mp=T,
                 end.time=as.POSIXct("2015-05-01 01:30:00.000", tz="UTC"),
                 volume.from=8.59, volume.to=8.72)
```
Using the same volume filtering approach, the next plot shows the behaviour of an individual market maker operating with a bid/ask spread limited between 3.63 and 3.83 bitcoin respectively. The rising bid prices at times are due to the event driven “leapfrog” phenomenon discussed previously.

```r
plotPriceLevels(lob.data$depth, price.from=235.65, price.to=237.65, volume.scale=10^-8, col.bias=1, start.time=as.POSIXct("2015-05-01 01:00:00.000", tz="UTC"), end.time=as.POSIXct("2015-05-01 03:00:00.000", tz="UTC"), volume.from=3.63, volume.to=3.83)
```
Liquidity

The `plotVolumePercentiles` function plots the available volume in 25bps increments on each side of the order book in the form of a stacked area graph.

The resulting graph is intended to display the market “quality” either side of the limit order book: The amount of volume available at increasing depths in the order book which would effect the VWAP (Volume Weighted Average Price) of a market order.

The example below favours buyers, since there is more volume available within 25 BPS of the best ask price in comparison to the thinner market -25 BPS below the best bid.

The top of the graph depicts the ask side of the book, whilst the bottom depicts the bid side. Percentiles and order book sides can be separated by an optional subtle line (`perc.line`) for improved legibility.

```r
plotVolumePercentiles(lob.data$depth.summary, volume.scale=10^-8, perc.line=F, 
start.time=as.POSIXct("2015-05-01 01:00:00.000", tz="UTC"), 
end.time=as.POSIXct("2015-05-01 04:00:00.000", tz="UTC"))
```
Zooming in to a 5 minute window and enabling the percentile borders with \texttt{(perc.line=FALSE)}, the amount of volume available at each 25 bps price level is exemplified.

\begin{verbatim}
# visualise 5 minutes of order book liquidity. # data will be aggregated to second-by-second resolution.
plotVolumePercentiles(lob.data$depth.summary,
    start.time=as.POSIXct("2015-05-01 04:30:00.000", tz="UTC"),
    end.time=as.POSIXct("2015-05-01 04:35:00.000", tz="UTC"),
    volume.scale=10^-8)
\end{verbatim}
Order cancellations

Visualising limit order cancellations can provide insights into order placement processes. The `plotVolumeMap()` function generates a visualisation of limit order cancellation events (excluding market and market limit orders).

```r
plotVolumeMap(lob.data$events, volume.scale=10^-8, log.scale = T)
```
Interestingly, the order cancellation visualisation shows the systematic activity of individual market participants. By filtering the display of cancellation events within a volume range, it is possible to isolate what are most likely individual order placement strategies. The following graph shows an individual strategy cancelling orders within the $[3.5, 4]$ volume range.

```r
plotVolumeMap(lob.data$events, volume.scale=10^-8, volume.from=3.5, volume.to=4)
```
Restricting the volume between [8.59, 8.72] shows a strategy placing orders at a fixed price at a fixed distance below the market price.

```r
plotVolumeMap(lob.data$events, volume.scale=10^-8, volume.from=8.59, volume.to=8.72)
```
Analysis

In addition to the visualisation functionality of the package, *obAnalytics* can also be used to study market event, trade and order book data.

Order book reconstruction

After loading and processing data, it is possible to reconstruct the limit order book for any given point in time. The next example shows the order book at a specific time (millisecond precision) limited to 10 price levels. The *liquidity* column shows the cumulative sum of volume from the best bid/ask up until each price level row.

```r
tp <- as.POSIXct("2015-05-01 04:25:15.342", tz="UTC")
ob <- orderBook(lob.data$events, max.levels=10)
print(ob)
```

<table>
<thead>
<tr>
<th>id</th>
<th>timestamp</th>
<th>liquidity</th>
<th>price</th>
<th>price</th>
<th>liquidity</th>
<th>timestamp</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>65619912</td>
<td>2015-05-01 05:03:13.566</td>
<td>0.16</td>
<td>235.45</td>
<td>235.71</td>
<td>3.91</td>
<td>2015-05-01 05:04:16.670</td>
<td>65620105</td>
</tr>
<tr>
<td>65620122</td>
<td>2015-05-01 05:04:26.395</td>
<td>1.10</td>
<td>235.12</td>
<td>235.71</td>
<td>7.70</td>
<td>2015-05-01 05:05:42.957</td>
<td>65620140</td>
</tr>
<tr>
<td>65620109</td>
<td>2015-05-01 05:04:18.993</td>
<td>2.03</td>
<td>235.10</td>
<td>235.72</td>
<td>7.91</td>
<td>2015-05-01 05:01:00.940</td>
<td>65619914</td>
</tr>
<tr>
<td>65618028</td>
<td>2015-05-01 05:00:08.424</td>
<td>4.57</td>
<td>235.01</td>
<td>235.80</td>
<td>21.11</td>
<td>2015-05-01 05:04:17.834</td>
<td>65620107</td>
</tr>
<tr>
<td>65619358</td>
<td>2015-05-01 04:54:21.109</td>
<td>4.67</td>
<td>234.95</td>
<td>235.81</td>
<td>34.31</td>
<td>2015-05-01 05:03:45.546</td>
<td>65620086</td>
</tr>
<tr>
<td>65598030</td>
<td>2015-05-01 00:39:56.799</td>
<td>4.80</td>
<td>234.92</td>
<td>235.84</td>
<td>50.29</td>
<td>2015-05-01 05:04:41.296</td>
<td>65620138</td>
</tr>
<tr>
<td>65620023</td>
<td>2015-05-01 05:02:33.711</td>
<td>5.88</td>
<td>234.74</td>
<td>235.85</td>
<td>73.11</td>
<td>2015-05-01 05:04:19.535</td>
<td>65620112</td>
</tr>
<tr>
<td>65597424</td>
<td>2015-05-01 00:23:05.230</td>
<td>28.36</td>
<td>234.54</td>
<td>236.05</td>
<td>86.75</td>
<td>2015-05-01 04:55:15.408</td>
<td>65619449</td>
</tr>
</tbody>
</table>

Market impacts

Using the *trades* and *events* data, it is possible to study market impact events.

All market impacts

The *tradeImpacts()* function groups individual trade events into market impact events. A market impact occurs when an order consumes 1 or more resting orders from the limit order book.

The following example shows the top 10 most aggressive sell impacts in terms of the *depth* removed from the order book in terms of BPS. The VWAP column indicates the volume weighted average price that the market *taker* received for their market order. In addition, the *hits* column shows the number of *hit* resting limit orders used to fulfil this market order.

```r
impacts <- tradeImpacts(lob.data$strades)
impacts <- impacts[impacts$dir == "sell", ]
bps <- 10000 * (impacts$max.price - impacts$min.price) / impacts$max.price
types <- with(lob.data, events$match(impacts$id, events$id), ]$type)
impacts <- cbind(impacts, type=types, bps)
head(impacts[order(-impacts$bps), ], 10)
```
### Individual impact

The main purpose of the package is to load limit order, quote and trade data for arbitrary analysis. It is possible, for example, to examine an individual market impact event. The following code shows the sequence of events as a single market (sell) order is filled.

The `maker.aggr` column shows how far each limit order was above or below the best bid when it was placed. The `age` column shows how long the order was resting in the order book before it was hit by this market order.

```r
impact <- with(lob.data, trades[trades$taker == 65596324, 
  c("timestamp", "price", "volume", "maker")])
makers <- with(lob.data, events[match(impact$maker, events$id), ])
makers <- makers[makers$action == "created", 
  c("id", "timestamp", "aggressiveness.bps")]
impact <- cbind(impact, maker=makers[match(impact$maker, makers$id), 
  c("timestamp", "aggressiveness.bps")])
age <- impact$timestamp - impact$maker.timestamp
impact <- cbind(impact[!is.na(age), c("timestamp", "price", "volume", 
  "maker.aggressiveness.bps"), age[!is.na(age)])
colnames(impact) <- c("timestamp", "price", "volume", "maker.aggr", "age")
impact$volume <- impact$volume*10^-8
print(impact)
```

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Price</th>
<th>Volume</th>
<th>Maker Aggr</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-05-01 00:11:01.533</td>
<td>235.09</td>
<td>0.21</td>
<td>16.62</td>
<td>0.891 secs</td>
</tr>
<tr>
<td>2015-05-01 00:11:01.563</td>
<td>234.70</td>
<td>1.00</td>
<td>5.54</td>
<td>3.331 secs</td>
</tr>
<tr>
<td>2015-05-01 00:11:01.592</td>
<td>234.70</td>
<td>1.03</td>
<td>0.00</td>
<td>1.094 secs</td>
</tr>
<tr>
<td>2015-05-01 00:11:01.657</td>
<td>234.42</td>
<td>3.81</td>
<td>-11.93</td>
<td>1.321 secs</td>
</tr>
<tr>
<td>2015-05-01 00:11:01.720</td>
<td>234.37</td>
<td>9.02</td>
<td>-13.64</td>
<td>6.730 secs</td>
</tr>
<tr>
<td>2015-05-01 00:11:01.786</td>
<td>234.35</td>
<td>3.68</td>
<td>-14.49</td>
<td>6.128 secs</td>
</tr>
<tr>
<td>2015-05-01 00:11:01.864</td>
<td>234.20</td>
<td>16.38</td>
<td>-20.88</td>
<td>5.552 secs</td>
</tr>
</tbody>
</table>

19