BTC Orderbook Trading Bot

Ronan Anderson

This file implements an algorithmic function, whereby any trading strategy can be used. In this file, We use an MA crossover just to test the function and demonstrate its performance.

We are using a set of three consecutive daily Bitcoin orderbooks from an exchange. This function will be built using the first two days and then backtested using the third day's data. We will also consider a trading fee of 1 basis point (or 0.01%) per trade and a trading latency of 100ms.

The orderbooks consist of the following: *MatchingTime : millisecond-precision timestamp of when the orderbook state was generated.* ReceivingTime : millisecond-precision timestamp of when a trading engine receives the state. *Symbol : the instrument (BTC).* Bid_Price_{i} : bid price in the orderbook. Bid_Price_1 is the highest bid price and Bid_Price_10 is the lowest bid price available in this orderbook. *Bid_Qty_{i} : bid quantity corresponding to each bid price, which means the amount of quantity on each price level.* Ask_Price_{i} : ask price in the orderbook. Ask_Price_1 is the lowest ask price and ASK_PRICE_10 is the highest ask price available in this orderbook. *Ask_Qty_{i} : ask quantity corresponding to each ask price, which means the amount of quantity on each price level.

1) Trading Strategy Contruction- Day 1 & 2

The 3 days are loaded in and labelled as follows:

```r
day1=read.csv("day1.csv.gz", header = FALSE, na.strings=c(""," ","NA"),
              stringsAsFactors=FALSE)
day2=read.csv("day2.csv.gz", header = FALSE, na.strings=c(""," ","NA"),
              stringsAsFactors=FALSE)
day3=read.csv("day3.csv.gz", header = FALSE, na.strings=c(""," ","NA"),
              stringsAsFactors=FALSE)
```

It is worth noting that day 2 picks up exactly where day 1 leaves off with regards to time. With this in mind, and with the trading strategy being formulated from both day 1 and day 2, we have combined these two files in the dataframe "newframe". The columns are then renamed respectively, and the date/time is formatted correctly. This would be useful if we were to plot any of the data. We largely concern ourselves with the order Receiving Time as this is the time that any data would be available to the company on the trading engine.

```r
newframe <- rbind(day1, day2)

colnames(newframe) <- c("Matching_Time", "Receiving_Time", "Symbol",
                        "Bid_Price_1", "Bid_Qty_1", "Ask_Price_1", "Ask_Qty_1",
                        "Bid_Price_2", "Bid_Qty_2", "Ask_Price_2", "Ask_Qty_2",
                        "Bid_Price_3", "Bid_Qty_3", "Ask_Price_3", "Ask_Qty_3",
                        "Bid_Price_4", "Bid_Qty_4", "Ask_Price_4", "Ask_Qty_4",
                        "Bid_Price_5", "Bid_Qty_5", "Ask_Price_5", "Ask_Qty_5",
                        "Bid_Price_6", "Bid_Qty_6", "Ask_Price_6", "Ask_Qty_6",
                        "Bid_Price_7", "Bid_Qty_7", "Ask_Price_7", "Ask_Qty_7",
                        "Bid_Price_8", "Bid_Qty_8", "Ask_Price_8", "Ask_Qty_8",
                           "Bid_Price_9", "Bid_Qty_9", "Ask_Price_9", "Ask_Qty_9",
                           "Bid_Price_10", "Bid_Qty_10", "Ask_Price_10", "Ask_Qty_10")

timeDate <- strptime(newframe$Receiving_Time, "%Y-%m-%dT%H:%M:%OS")
op <- options(digits.secs = 3)
newframe$Receiving_Time <- timeDate
```

The trading strategy implemented here is a Simple Moving Average Crossover. The moving average periods used are up to interpretation. Because of the frequency of this orderbook being so high with 30 to 40 orders per second, we will use 60,000 for the long term MA and 15,000 for a short term MA. This would give a long term MA covering the approximately the previous half hour of orders, which may represent some form of longer term average.

Note that there is no evidence behind this strategy and it is not one that we would implement in a real-life scenario. It is being used because it is relatively simple and allows us to test this signal function appropriately.

Next, we create a dataframe with the relevant information. We use the lowest bid price and the highest ask price so that our strategy is more robust and does not rely on generous market spreads. We also lead both the bid and ask prices by 35 in order to combat the latency of 100ms (approximately 30 to 40 trades per 100ms).

```
short <- 15000
long <- 60000
df_ema <- data.frame('Time' = newframe$Receiving_Time,
                     'Bid' = lead(newframe$Bid_Price_10, 35),
                     'Ask' = lead(newframe$Ask_Price_10, 35),
                     'SMAshort' = SMA(newframe$Bid_Price_10, n=short),
                     'SMAlong' = SMA(newframe$Bid_Price_10, n=long))
```

With the moving averages calculated, we can implement the trading strategy as follows, where we buy BTC when the short-term MA crosses above the long-term MA, and sell when it crosses under.

```
df_ema[long:length(newframe$Bid_Price_10),] %>%
  mutate(
    signal = case_when(
      SMAshort >= SMAlong ~ "buy",
      SMAshort < SMAlong ~ "sell"
    ),
    previous_signal = lag(signal, 1),
    decision = case_when(
      signal == previous_signal ~ "hold",
      TRUE ~ signal
    )
  ) -> df_with_decisions

df_with_decisions %>%
  filter(decision != 'hold') %>%
  select(Time, Bid, Ask, SMAshort, SMAlong, decision) -> df_bought_sold

head(df_bought_sold, n=4)
```

```
##                      Time   Bid   Ask SMAshort  SMAlong decision
## 1 2021-05-19 00:39:45.289 42822 42849 42909.05 43198.12     sell
## 2 2021-05-19 02:40:56.767 41109 41149 40947.69 40947.69      buy
## 3 2021-05-19 02:52:31.892 40762 40790 40939.37 40939.38     sell
## 4 2021-05-19 03:21:52.750 41001 41034 40687.16 40687.16      buy
```

This data frame contains our trading signals, the Bid and Ask price, and the time that the signal is generated.

Lastly, we can backtest the trading strategy with $100 as given. Note that the transaction fee of 1bp is also accounted for in the "amount" variable. The transactions and their values are stored in the dataframe "df_action". This strategy is a long only strategy, and therefore if a sell signal arises as the first signal, its transaction fee is ignored and it is removed from the data frame "df_action".

```r
total <- 100
base <- 100
df_action <- data.frame(Action = NA)
for(i in 1:nrow(df_bought_sold)){

  if(df_bought_sold[i,]$decision == "buy"){

    price_bought <- df_bought_sold[i,]$Ask # takes ask price at time of signal
    if(i == 2){total <- base} # resets total to 100 if first signal is to sell
    amount <- total - (total * 0.0001) # 1bp (0.01%) transaction taken from current total
    shares <- amount/price_bought
    total <- price_bought * shares
    df_action[i, 'Action'] <- paste0("BUY: ", round(shares, digits=7), " shares at $",
                                     price_bought, " for $", round(total, 4))

  } else if(total > 0 && df_bought_sold[i,]$decision == "sell"){

    price_sold <- df_bought_sold[i,]$Bid
    if(i == 1){shares <- base/price_sold}
    total <- shares * price_sold
    total <- total - (total * 0.0001)
    df_action[i, 'Action'] <- paste0("SELL: ", round(shares, digits=7), " shares at $",
                                     price_sold, " for $", round(total, 4))

  }
}

if(df_bought_sold[1,]$decision == "sell"){df_action <- df_action[-1,]}

head(df_action, n=3)
```

```
## [1] "BUY: 0.0024299 shares at $41149 for $99.99"
## [2] "SELL: 0.0024299 shares at $40762 for $99.0397"
## [3] "BUY: 0.0024134 shares at $41034 for $99.0298"
```

```r
cat("\n")
```

```r
tail(df_action, n=3)
```

```
## [1] "BUY: 0.0022062 shares at $41670 for $91.9325"
## [2] "SELL: 0.0022062 shares at $41088 for $90.6394"
## [3] "BUY: 0.0022145 shares at $40925 for $90.6303"
```

```r
cat("\n")
```

```r
total_return <- total/base
print(paste0("Total Return: ", round(total_return*100, 3)-100, "%"))
```

```
## [1] "Total Return: -9.37%"
```

By trading across the 48 hour period of day 1 and day 2, the strategy has executed 94 trades for a return of

-9.37%.

2) Trading Strategy Testing - Day 3

We will now embed this strategy into a function called "maco_return" so that it can be easily called and tested on the day 3 data, or any future data. The only requirements are that the orderbooks follow the same format as previous.

The inputs to this function are: *data - the orderbook to run the function on.* short - the short term MACO length. *long - the long term MACO length.* total - the total amount in the trading account. *base - the base amount that the trading account began with (same as total at beginning).

```r
maco_return <- function(data, short, long, total, base) {

  colnames(data) <- c("Matching_Time", "Receiving_Time", "Symbol",
                        "Bid_Price_1", "Bid_Qty_1", "Ask_Price_1", "Ask_Qty_1",
                        "Bid_Price_2", "Bid_Qty_2", "Ask_Price_2", "Ask_Qty_2",
                        "Bid_Price_3", "Bid_Qty_3", "Ask_Price_3", "Ask_Qty_3",
                        "Bid_Price_4", "Bid_Qty_4", "Ask_Price_4", "Ask_Qty_4",
                        "Bid_Price_5", "Bid_Qty_5", "Ask_Price_5", "Ask_Qty_5",
                        "Bid_Price_6", "Bid_Qty_6", "Ask_Price_6", "Ask_Qty_6",
                        "Bid_Price_7", "Bid_Qty_7", "Ask_Price_7", "Ask_Qty_7",
                        "Bid_Price_8", "Bid_Qty_8", "Ask_Price_8", "Ask_Qty_8",
                        "Bid_Price_9", "Bid_Qty_9", "Ask_Price_9", "Ask_Qty_9",
                        "Bid_Price_10", "Bid_Qty_10", "Ask_Price_10", "Ask_Qty_10")

  timeDate <- strptime(data$Receiving_Time, "%Y-%m-%dT%H:%M:%OS")
  op <- options(digits.secs = 3)
  data$Receiving_Time <- timeDate

  df_ema <- data.frame('Time' = data$Receiving_Time,
                       'Bid' = lead(data$Bid_Price_10, 35),
                       'Ask' = lead(data$Ask_Price_10, 35),
                       'SMAshort' = SMA(data$Bid_Price_10, n=short),
                       'SMAlong' = SMA(data$Bid_Price_10, n=long))


  df_ema[long:length(data$Bid_Price_10),] %>%
    mutate(
      signal = case_when(
        SMAshort >= SMAlong ~ "buy",
        SMAshort < SMAlong ~ "sell"
      ),
      previous_signal = lag(signal, 1),
      decision = case_when(
        signal == previous_signal ~ "hold",
        TRUE ~ signal
      )
    ) -> df_with_decisions

  df_with_decisions %>%
    filter(decision != 'hold') %>%
    select(Time, Bid, Ask, SMAshort, SMAlong, decision) -> df_bought_sold

  head(df_bought_sold)
  df_action <- data.frame(Action = NA)
  for(i in 1:nrow(df_bought_sold)){

    if(df_bought_sold[i,]$decision == "buy"){

    price_bought <- df_bought_sold[i,]$Ask
    if(i == 2){total <- base} #required in case first trade signal is sell.
    amount <- total - (total * 0.0001)
    shares <- amount/price_bought
    total <- price_bought * shares
    df_action[i, 'Action'] <- paste0("BUY: ", round(shares, digits=7), " shares at $",
                                     price_bought, " for $", round(total, 4))

  } else if(total > 0 && df_bought_sold[i,]$decision == "sell"){

    price_sold <- df_bought_sold[i,]$Bid
    if(i == 1){shares <- base/price_sold}
    total <- shares * price_sold
    total <- total - (total * 0.0001)
    df_action[i, 'Action'] <- paste0("SELL: ", round(shares, digits=7), " shares at $",
                                     price_sold, " for $", round(total, 4))

  }
}

  if(df_bought_sold[1,]$decision == "sell"){df_action <- df_action[-1,]}

  total_return <- total/base
  return(print(paste0("Total Return: ", round(total_return*100, 3)-100, "%")))
}


maco_return(day3, 15000, 60000, 100, 100)
```

```
## [1] "Total Return: -12.474%"
```