FE570 - Homework #2

I pledge my honor that I have abided by the Stevens Honor System.

Sid Bhatia

2023-10-16

Problem 2.1

```
# Load necessary packages.
library(xts)
```

1.

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## ###################################### WARNING ######################################
## # We noticed you have dplyr installed. The dplyr lag() function breaks how    #
## # base R's lag() function is supposed to work, which breaks lag(my_xts).      #
## #                                                                             #
## # If you call library(dplyr) later in this session, then calls to lag(my_xts) #
## # that you enter or source() into this session won't work correctly.          #
## #                                                                             #
## # All package code is unaffected because it is protected by the R namespace   #
## # mechanism.                                                                  #
## #                                                                             #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## # You can use stats::lag() to make sure you're not using dplyr::lag(), or you #
## # can add conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop   #
## # dplyr from breaking base R's lag() function.                                #
## ###################################### WARNING ######################################
```

```
library(highfrequency)

# Load in data set.
options(digits.secs=3)
absolute_path <- 'C:/Users/sbhatia2/My Drive/University/Academics/Semester V/FE570 - Market Microstructu
load(paste(absolute_path, "sampleTQdata.RData", sep = ""))

# Added to remove warnings about time zone mismatch.
Sys.setenv(TZ='GMT')
```

```
head(tqdata)
```

```
##                      SYMBOL EX     BID BIDSIZ     OFR OFRSIZ MODE   PRICE SIZE
## 2008-01-04 09:30:27    XXX  N 193.340    4.5 193.890   11.5   12 193.710 9100
## 2008-01-04 09:30:28    XXX  N 193.340    4.5 193.890   11.5   12 193.590  200
## 2008-01-04 09:30:29    XXX  N 193.250   12.5 193.810    8.5   12 193.445  200
## 2008-01-04 09:30:30    XXX  N 193.470    0.5 193.630    0.5   12 193.380  250
## 2008-01-04 09:30:31    XXX  N 193.470    0.5 193.630    0.5   12 193.340  300
## 2008-01-04 09:30:33    XXX  N 193.300    2.5 193.640    0.5   12 193.520  400
```

```
tail(tqdata)
```

```
##                      SYMBOL EX     BID BIDSIZ     OFR OFRSIZ MODE   PRICE SIZE
## 2008-01-04 15:59:52    XXX  N 191.600   60.5 191.670    3.5   12 191.695  550
## 2008-01-04 15:59:55    XXX  N 191.620    0.5 191.790    1.5   12 191.620 1600
## 2008-01-04 15:59:57    XXX  N 191.600    180 191.690   27.5   12 191.690  350
## 2008-01-04 15:59:58    XXX  N 191.600    180 191.690   27.5   12 191.650  150
## 2008-01-04 15:59:59    XXX  N 191.600    180 191.690   27.5   12 191.620   50
## 2008-01-04 16:00:00    XXX  N 191.600    180 191.690   27.5   12 191.670   50
```

i. How many trades are in the dataset?

```
# Retrieve number of trades by counting number of rows in dataset.
num_of_trades <- nrow(tqdata)
num_of_trades
```

```
## [1] 8153
```

As seen above, there are 8153 trades in the dataset.

ii. Plot the trade prices p_t and the best-bid b_t and best-ask prices a_t for the entire dataset.

```
# Convert dataset to data frame for easier access.
TQ_df <- data.frame(Date = index(tqdata), tqdata)
head(TQ_df)
```

```
##                                    Date SYMBOL EX     BID BIDSIZ     OFR OFRSIZ
## 2008-01-04 09:30:27 2008-01-04 09:30:27    XXX  N 193.340    4.5 193.890   11.5
## 2008-01-04 09:30:28 2008-01-04 09:30:28    XXX  N 193.340    4.5 193.890   11.5
## 2008-01-04 09:30:29 2008-01-04 09:30:29    XXX  N 193.250   12.5 193.810    8.5
## 2008-01-04 09:30:30 2008-01-04 09:30:30    XXX  N 193.470    0.5 193.630    0.5
## 2008-01-04 09:30:31 2008-01-04 09:30:31    XXX  N 193.470    0.5 193.630    0.5
## 2008-01-04 09:30:33 2008-01-04 09:30:33    XXX  N 193.300    2.5 193.640    0.5
##                      MODE   PRICE SIZE
## 2008-01-04 09:30:27   12 193.710 9100
## 2008-01-04 09:30:28   12 193.590  200
## 2008-01-04 09:30:29   12 193.445  200
## 2008-01-04 09:30:30   12 193.380  250
## 2008-01-04 09:30:31   12 193.340  300
## 2008-01-04 09:30:33   12 193.520  400
```

```
# Retrieve the 'asks' or 'offers' from data frame.
asks <- as.numeric(TQ_df$OFR)
head(asks)
```

```
## [1] 193.89 193.89 193.81 193.63 193.63 193.64
```

```r
# Retrieve 'bids' from data frame.
bids <- as.numeric(TQ_df$BID)
head(bids)
```

```
## [1] 193.34 193.34 193.25 193.47 193.47 193.30
```

```r
# Compute the 'mid' or middle price between bid and ask.
mids <- (bids + asks) * 0.5
head(mids)
```

```
## [1] 193.615 193.615 193.530 193.550 193.550 193.470
```

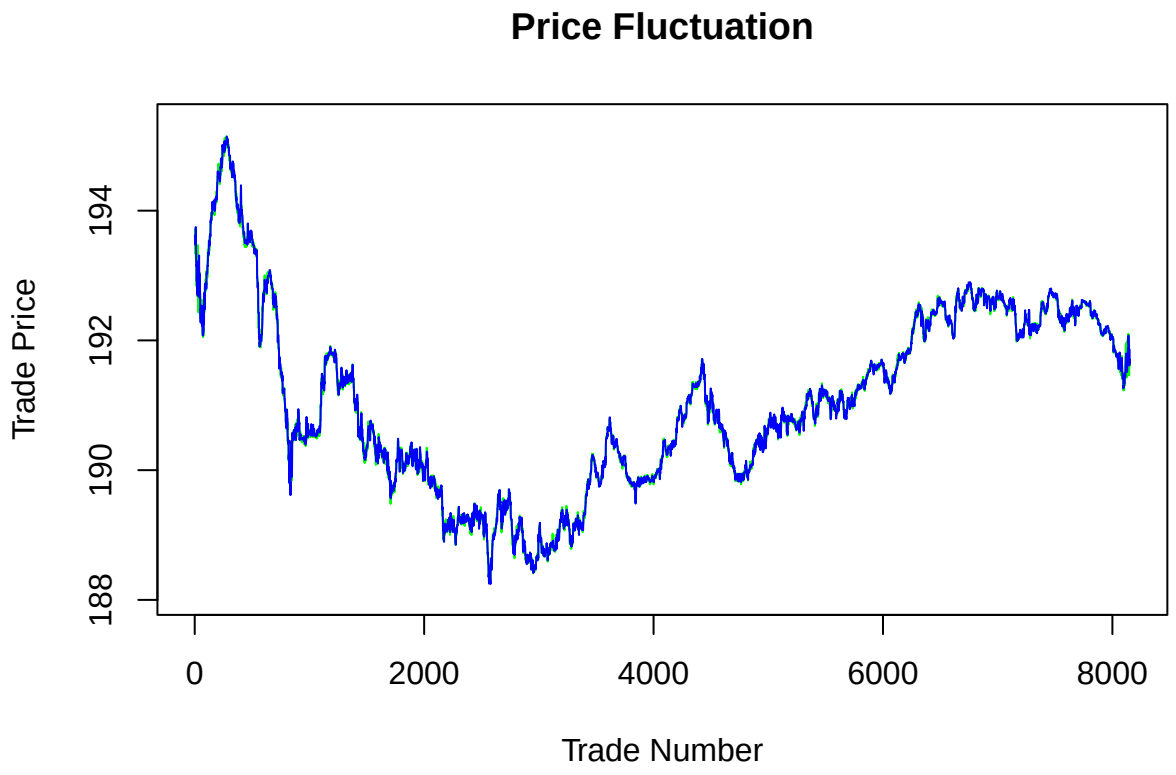
```r
# Establish minimum and maximum prices quoted.
p_min <- min(as.numeric(TQ_df$PRICE))
p_max <- max(as.numeric(TQ_df$PRICE))

p_min
```

```
## [1] 188.26
```

```r
p_max
```

```
## [1] 195.15
```

```r
# Plot trade prices and respective best-bid and best-ask prices using the mid price.
plot(as.numeric(TQ_df$PRICE), col = "green", type = "l", ylab = "Trade Price",
    xlab = "Trade Number", main = "Price Fluctuation", ylim = c(p_min - 0.2, p_max + 0.2))
lines(mids, type = "l", col = "blue")
```



iii. Do the same as in ii) but only for trades with counts 100:200 (100th trade to 200th trade).

```
# Retrieve sliced data frame from 100 to 200 trades.
TQ_df_sliced <- data.frame(tqdata[100:200])
head(TQ_df_sliced)
```

```
##                      SYMBOL EX     BID BIDSIZ     OFR OFRSIZ MODE   PRICE SIZE
## 2008-01-04 09:33:18    XXX  N 192.790      6 192.970    1.5   12 192.910  850
## 2008-01-04 09:33:19    XXX  N 192.760      3 192.830    1.5   12 192.900   50
## 2008-01-04 09:33:20    XXX  N 192.810    0.5 192.970    0.5   12 192.850   50
## 2008-01-04 09:33:32    XXX  N 192.950      2 193.110    2.5   12 193.070  100
## 2008-01-04 09:33:36    XXX  N 192.950      2 193.110    2.5   12 192.990   50
## 2008-01-04 09:33:39    XXX  N 193.020    0.5 193.070      1   12 193.070  100
```

```
# Retrieve the 'asks' or 'offers' from data frame.
asks_2 <- as.numeric(TQ_df_sliced$OFR)
head(asks_2)
```

```
## [1] 192.97 192.83 192.97 193.11 193.11 193.07
```

```
# Retrieve 'bids' from data frame.
bids_2 <- as.numeric(TQ_df_sliced$BID)
head(bids_2)
```

```
## [1] 192.79 192.76 192.81 192.95 192.95 193.02
```

```
# Compute the 'mid' or middle price between bid and ask.
mids_2 <- (bids_2 + asks_2) * 0.5
head(mids_2)
```

```
## [1] 192.880 192.795 192.890 193.030 193.030 193.045
```

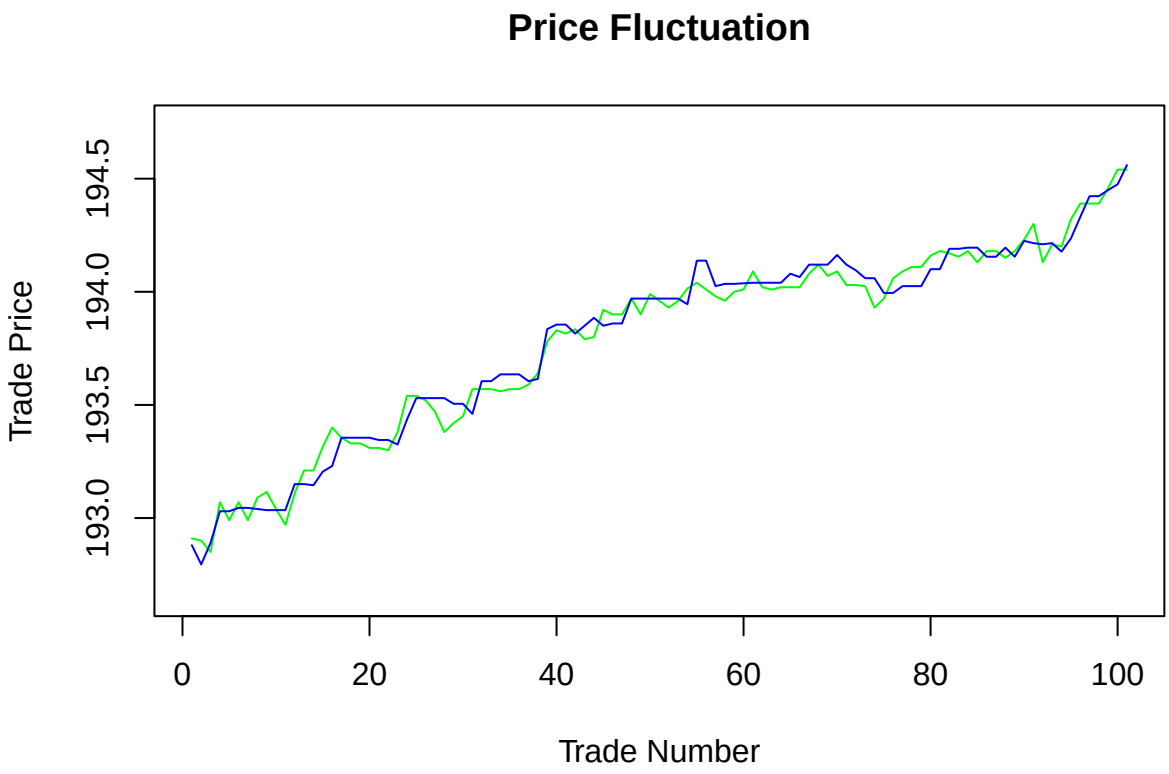
```
# Establish minimum and maximum prices quoted.
p_min_2 <- min(as.numeric(TQ_df_sliced$PRICE))
p_max_2 <- max(as.numeric(TQ_df_sliced$PRICE))

p_min_2
```

```
## [1] 192.85
```

```
p_max_2
```

```
## [1] 194.54
```

```
# Plot trade prices and respective best-bid and best-ask prices using the mid price.
plot(as.numeric(TQ_df_sliced$PRICE), col = "green", type = "l", ylab = "Trade Price",
    xlab = "Trade Number", main = "Price Fluctuation", ylim = c(p_min_2 - 0.2, p_max_2 + 0.2))
lines(mids_2, type = "l", col = "blue")
```



2. Count how many trades take place within the spread ($p_t \in (b_t, a_t)$), and how many at the touch ($p_t = b_t$ or $p_t = a_t$). Give separately the three numbers, and test if their sum reproduces the total trade count from 1.i).

```
bid <- sapply(TQ_df$BID, FUN = as.numeric)
ask <- sapply(TQ_df$OFR, FUN = as.numeric)
price <- sapply(TQ_df$PRICE, FUN = as.numeric)

# Check for number of prices within the range.
within <- length( which( (price > bid) & (price < ask) ) )

# Check for number of prices outside the range.
outside <- length( which ( (price < bid) | (price > ask) ) )

# Check for number of prices at the bid.
at_bid <- length( which( price == bid ) )

# Check for number of prices at the offer.
at_offer <- length( which( price == ask) )

within
```

```
## [1] 2832
```

```
outside
```

```
## [1] 2242
```

```
at_bid
```

```
## [1] 1709
```

```
at_offer
```

```
## [1] 1370
```

```
# Check if sum equals the total number of trades (8153).
sum(within, outside, at_bid, at_offer) == num_of_trades
```

```
## [1] TRUE
```

3. Determine the "trade direction" d_t of each trade, which shows if it is a buy ($d_t = +1$) or if it is a sell ($d_t == -1$).

Implement each of the following ways:

i. Tick Test: Use only the trade prices p_t, but not the quotes a_t and b_t. Under the test, the trade is classified as a buy/sell according to: - $d_t = +1$ (buy) if $p_t > p_{t-1}$ (uptick) or if $p_t = p_{t-1} > p_{t-2}$ (zero-uptick) - $d_t = -1$ (sell) if $p_t < p_{t-1}$ (downtick) or if $p_t = p_{t-1} < p_{t-2}$ (zero-downtick)

Note that zero-uptick/downtick results apply also if there are multiple (more than 2) trades with the same price.

For example if the trade prices are $p_t = (19.9, 20.0, 20.0, 20.0)$ (increasing t order), then the trade signs are (?, +, + , +).

```
# Create a function that implements the Tick Test.
tick_test <- function(price)
{
    sign <- c(1)
    for(i in 2:(length(price)))
    {
        if(price[i] < price[i - 1])
        {
            sign <- c(sign, -1)
        }
        else if(price[i] > price[i - 1])
        {
            sign <- c(sign, 1)
        }
        else
        {
            sign <- c(sign, sign[i - 1])
        }
    }
    return(sign)
}
```

ii. Lee-Ready Rule: Use both p_t and quotes a_t and b_t. The Lee-Ready Rule decides if a trade is a buy or sell by comparing the trade price p_t with the mid-price $m_t = \frac{1}{2}(a_t + b_t)$ (the half-point between best-bid b_t and best-ask a_t).

If the trade price is exactly equal to the mid-price, $p_t = m_t$, then use the tick rule in point (i) above.

```
# Create a function that implements the Lee-Ready Rule.
lee_ready_rule <- function(price)
```

```
{
    tick <- tick_test(price)
    sign <- c(1)
    bid <- sapply(TQ_df$BID, FUN = as.numeric)
    ask <- sapply(TQ_df$OFR, FUN = as.numeric)

    for(i in 2:(length(price)))
    {
        mid <- (bid[i] + ask[i]) * 0.5

        if(price[i] > mid)
        {
            sign <- c(sign, 1)
        }
        else if(price[i] < mid)
        {
            sign <- c(sign, -1)
        }
        else
        {
            sign <- c(sign, tick[i])
        }
    }
    return(sign)
}
```

```
# Apply custom functions vs. library functions.
Tick_Test_TQ <- tick_test(price)
Lee_Ready_Rule_TQ <- lee_ready_rule(price)

Lee_Ready_Rule_Actual <- getTradeDirection(tqdata)

# Check to see if Lee-Ready implementation is the same.
length( which(Lee_Ready_Rule_Actual == Lee_Ready_Rule_TQ ) ) / length(Lee_Ready_Rule_TQ)
```

```
## [1] 1
```

```
# Check to see difference between Tick Rule classification and Lee-Ready.
length( which(Tick_Test_TQ ==  Lee_Ready_Rule_TQ) ) / length(Tick_Test_TQ)
```

```
## [1] 0.7944315
```

As a result, approximately 79.4% of the trades are classified as the same way according to the two different methodologies, Tick Test and Lee-Ready Rule.

As seen above, the custom made function 'Lee_Ready_Rule_Actual' classifies the trades the same way as the in-built function 'getTradeDirection'.