# Stock Analysis with Donald Trump Tweets

Harsh, Deepanshu Tyagi May 7, 2017

## STEP 1: Extract company names from Donald Trump's Tweets

First, we will load all of the libraries neccessary to conduct this analysis.

```
library(plyr);library(dplyr);library(readr);library(twitteR);library(streamR)
;library(ROAuth);library(RCurl);library(stringr);library(stringdist);library(
pbapply);library(readxl);library(PerformanceAnalytics);library(xts);library(g
data);library(lubridate);library(tidyr);library(tidytext);
```

Next, read in authentication credentials to allow us to connect to the Twitter API.

```
twitter_cred <- read.csv("C:/Users/mnest/Google
Drive/RWD/Twitter/twitter_credentials.csv")
my_oauth <-
setup_twitter_oauth(twitter_cred[1,2],twitter_cred[2,2],twitter_cred[3,2],twi
tter_cred[4,2])
```

Get Tweets from the official Donald Trump Twitter account (@realDonaldTrump)

```
trump <- getUser("@realDonaldTrump")
#Tweets from API are stored in a list where each element of the list is an
individual tweet
trump_tweets <- userTimeline(trump, n=2000)
#Convert list of Tweets into a DataFrame
trump_tweets_df <- twListToDF(trump_tweets)
```

Read in all of the publicly traded company names from NASDAQ and NYSE. Remove common words from company names to increase accuracy of string distance computations.

```
nasdaq <- read_csv("C:/Users/mnest/Google Drive/Stevens/FE-
582(DataScience)/group_project/nasdaq.csv")
nyse <- read_csv("C:/Users/mnest/Google Drive/Stevens/FE-
582(DataScience)/group_project/nyse.csv")
stopwords <- "\\b(Corporation|Inc|Inc.|Corp|Co|Group|Resources|Systems)\\b"
nasdaq$Name_clean <- str_trim(str_replace_all(nasdaq$Name, stopwords, ""))
nyse$Name_clean <- str_trim(str_replace_all(nyse$Name, stopwords, ""))
```

Define a function to read each Trump tweet, strip all of the words in the tweet and compare them to the lost of publicly traded company names. If the string similarity is within a predetermined threshold, then we accept this as a company that trump has mentioned in his tweet.

```
get_trump_tweet_companies <- function(tweet){
     tmp_split <- unlist(str_split(toupper(iconv(enc2utf8(tweet))), " "))

     trump_nyse_companies <- sapply(tmp_split, function(x) max(1 -
stringdist(x, toupper(nyse$Name_clean), method = "jw", p=.2))>.98)
     trump_nyse_companies_list <-
sapply(names(trump_nyse_companies)[trump_nyse_companies], function(x)
nyse$Name[amatch(x, nyse$Name_clean, method = "jw", p=.2, maxDist = 2000)])

     trump_nasdaq_companies <- sapply(tmp_split, function(x) max(1 -
stringdist(x, toupper(nasdaq$Name_clean), method = "jw", p=.2))>.98)
     trump_nasdaq_companies_list <-
sapply(names(trump_nasdaq_companies)[trump_nasdaq_companies], function(x)
nasdaq$Name[amatch(x, nasdaq$Name_clean, method = "jw", p=.2, maxDist =
2000)])

     results <- na.omit(c(trump_nyse_companies_list,
trump_nasdaq_companies_list))
     names(results) <- NULL

     return(unlist(results)[1])
}
```

Create a list that will contain all of the companies that trump has mentioned in his tweets.

```
company_mentions <- pblapply(trump_tweets_df$text, get_trump_tweet_companies)
#Append this information to our original Trump Twitter DataFrame
trump_tweets_df$company_mentions <- company_mentions
#Append a logical column indicating if a given tweet contains a company name.
trump_tweets_df$is_populated <- lengths(trump_tweets_df$company_mentions) >=
1
```

## STEP 2: Analyze the volatility of each stocks Trump has mentioned in his tweets.

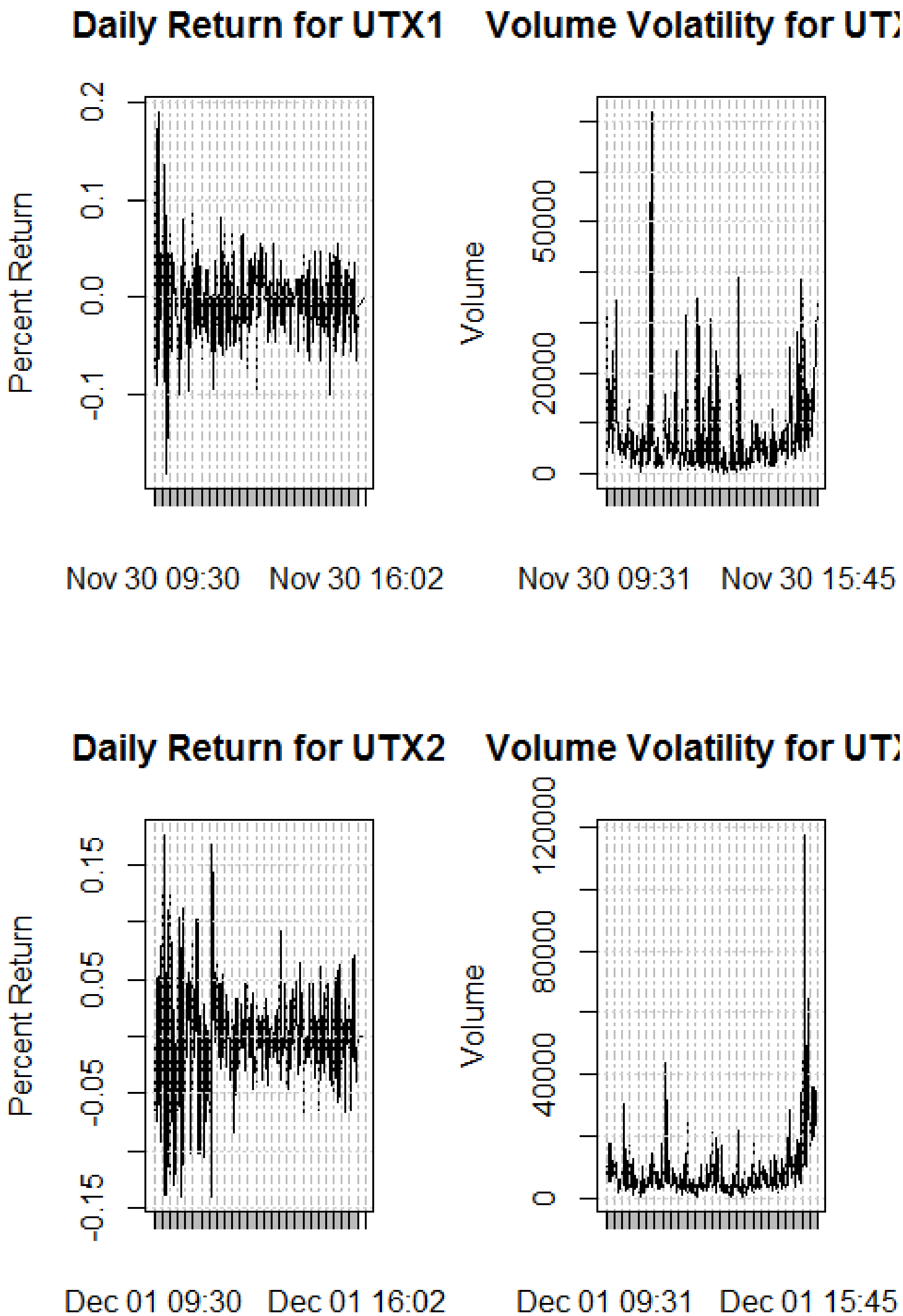
After the data was acquired from the Bloomberg terminal, we created an excel spreadsheet where each sheet contains information from the time when Trump mentioned a company. Then, we defined a function to read each sheet and plot the time series of respective Percent Return and Volume.

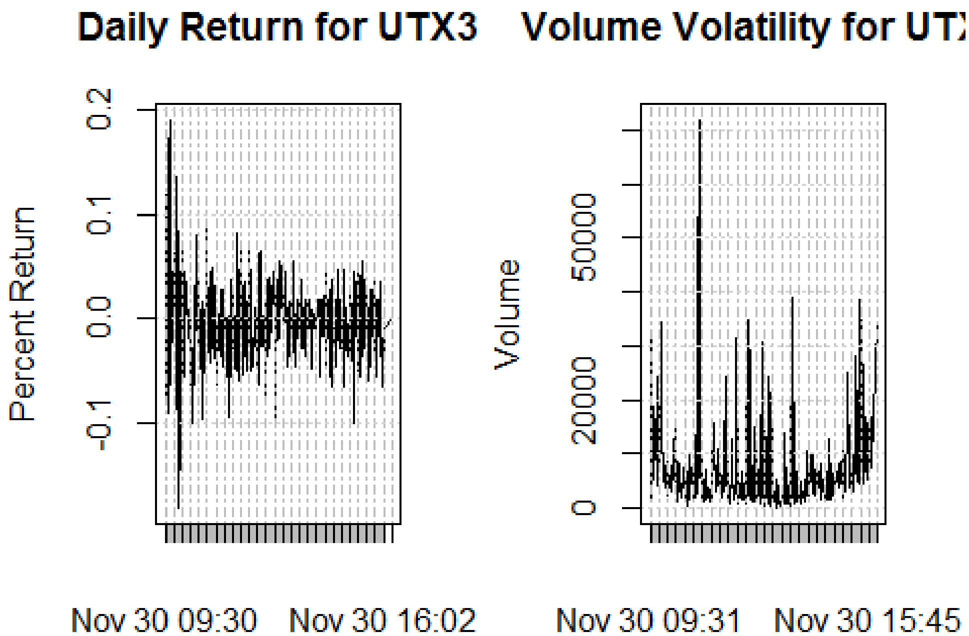
```
get_volatility_analysis <- function(trump_tweet_mention){
     trump_min_df <- read_excel("C:/Users/neste/Google Drive/Stevens/FE-
582(DataScience)/group_project/trump_minute_data.xlsx",sheet =
```

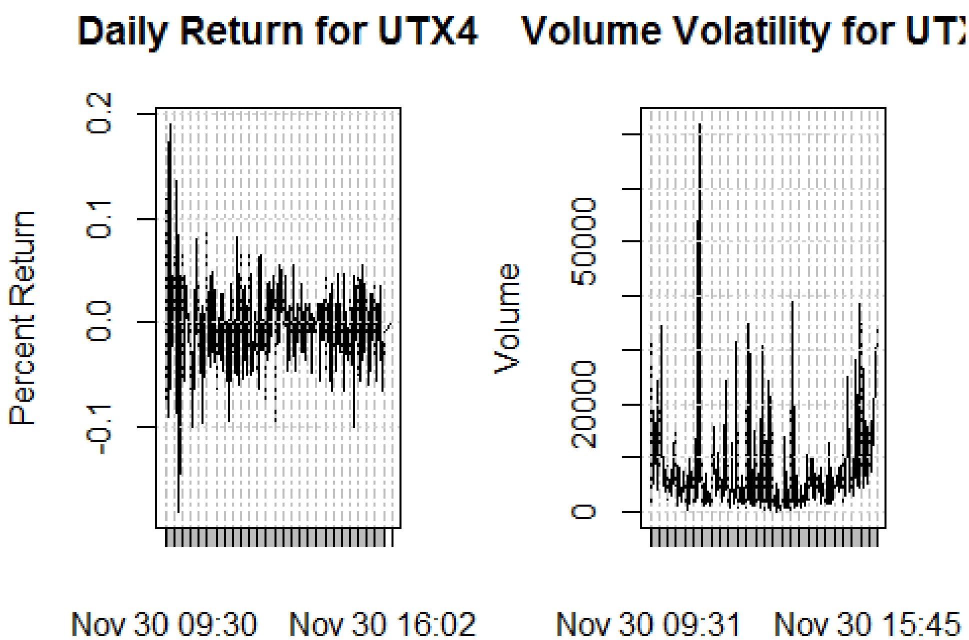
```
trump_tweet_mention, skip = 2)
     day_boundary=grep("9:30",trump_min_df$Date)

     day1=trump_min_df[1:day_boundary[2]-1, c("Date","LAST_PRICE", "VOLUME")]

     #Volume outlier detection, removes the first record, and all records
following 3:59pm
     remove_me <- -grep(x = day1$Date, pattern = "(9[:]30|(
16[:])|(15[:]59))")

     day1_xts <- xts(day1$LAST_PRICE, order.by = day1$Date)

     day1_returns <- CalculateReturns(day1_xts)

     day1_return_pct <- day1_returns * 100

     day1_volume <- xts(day1$VOLUME[remove_me], order.by =
day1$Date[remove_me])

     trump_tweet <- unlist(read_excel("C:/Users/neste/Google
Drive/Stevens/FE-582(DataScience)/group_project/trump_minute_data.xlsx",sheet
= trump_tweet_mention)[1,8]);names(trump_tweet) <- NULL
     trump_tweet <- gsub(trump_tweet, pattern = "(AM|PM)", replacement = "")

     par(mfrow = c(1,2))
     plot(day1_return_pct, main = sprintf("Daily Return for %s",
trump_tweet_mention), ylab = "Percent Return")
     plot(day1_volume, main = sprintf("Volume Volatility for %s",
trump_tweet_mention), ylab = "Volume")

}
```

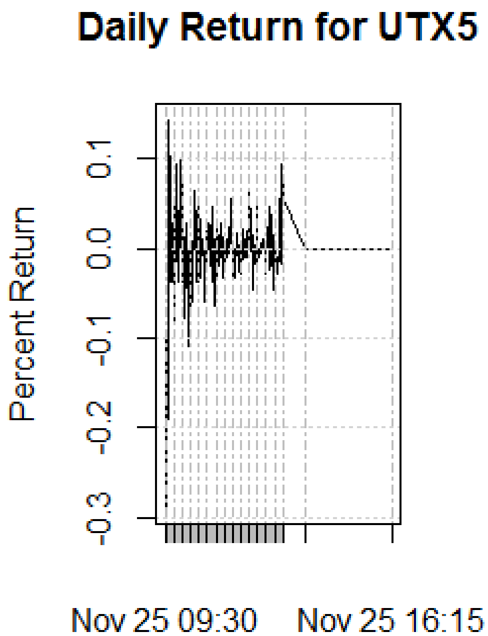
Next, we establish a character vector containint the tickers used in each sheet of our data, then run the function to generate the volatility plots.

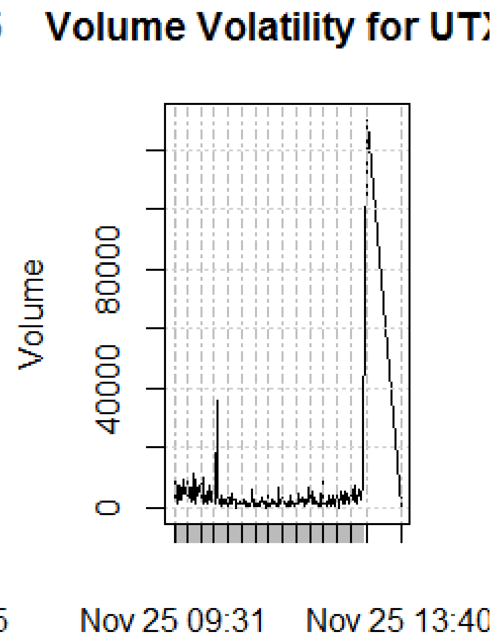
```
#Character vector of tickers Trump mentioned in his tweets
trump_stock_mentions <- c("UTX1", "UTX2", "UTX3", "UTX4", "UTX5", "RXN",
"BA1", "BA2", "F1", "F2", "F3", "F4", "F5", "TWTR", "GOOG", "FB")
#For loop to return plots of each sheet
for(x in trump_stock_mentions){
     get_volatility_analysis(x)
}
```

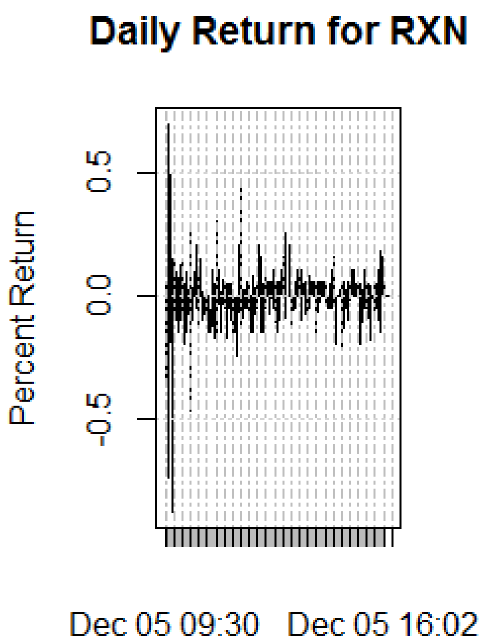

Daily Return for UTX1
Volume Volatility for UT
Percent Return
0.2
0.1
0.0
-0.1
Nov 30 09:30
Nov 30 16:02
Volume
50000
20000
0
Nov 30 09:31
Nov 30 15:45
Daily Return for UTX2
Volume Volatility for UT
Percent Return
0.15
0.05
-0.05
-0.15
Dec 01 09:30
Dec 01 16:02
Volume
120000
80000
40000
0
Dec 01 09:31
Dec 01 15:45

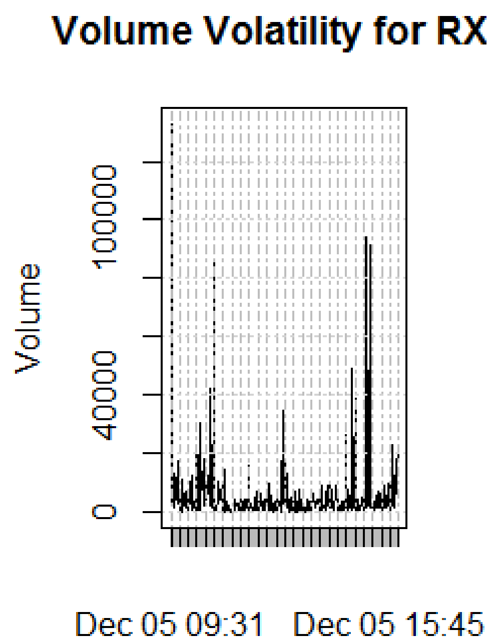

Daily Return for UTX3
Volume Volatility for UT
Percent Return
0.2
0.1
0.0
-0.1
Nov 30 09:30
Nov 30 16:02
Volume
50000
20000
0
Nov 30 09:31
Nov 30 15:45

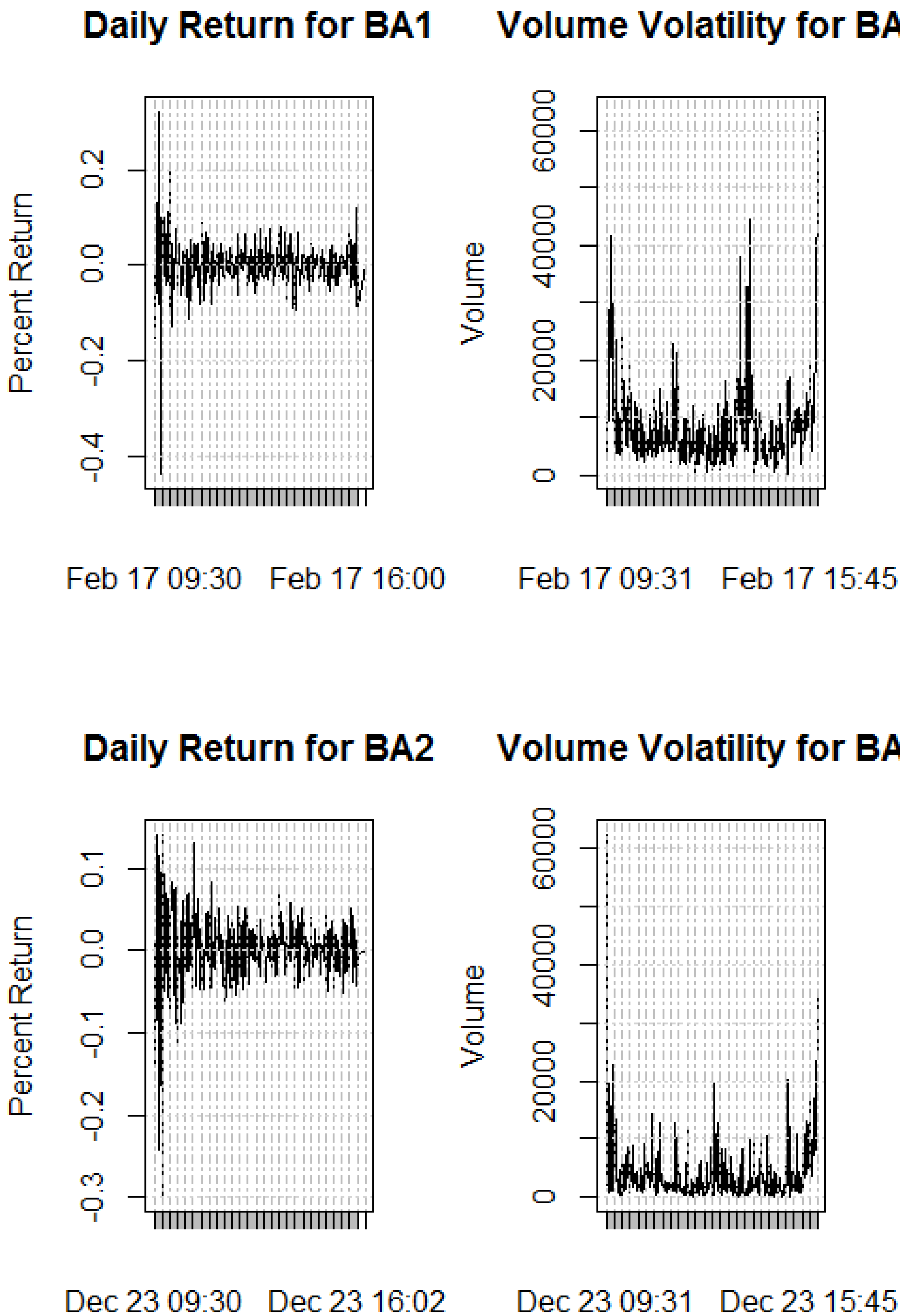

Daily Return for UTX4
Volume Volatility for UT
Percent Return
0.2
0.1
0.0
-0.1
Nov 30 09:30
Nov 30 16:02
Volume
50000
20000
0
Nov 30 09:31
Nov 30 15:45

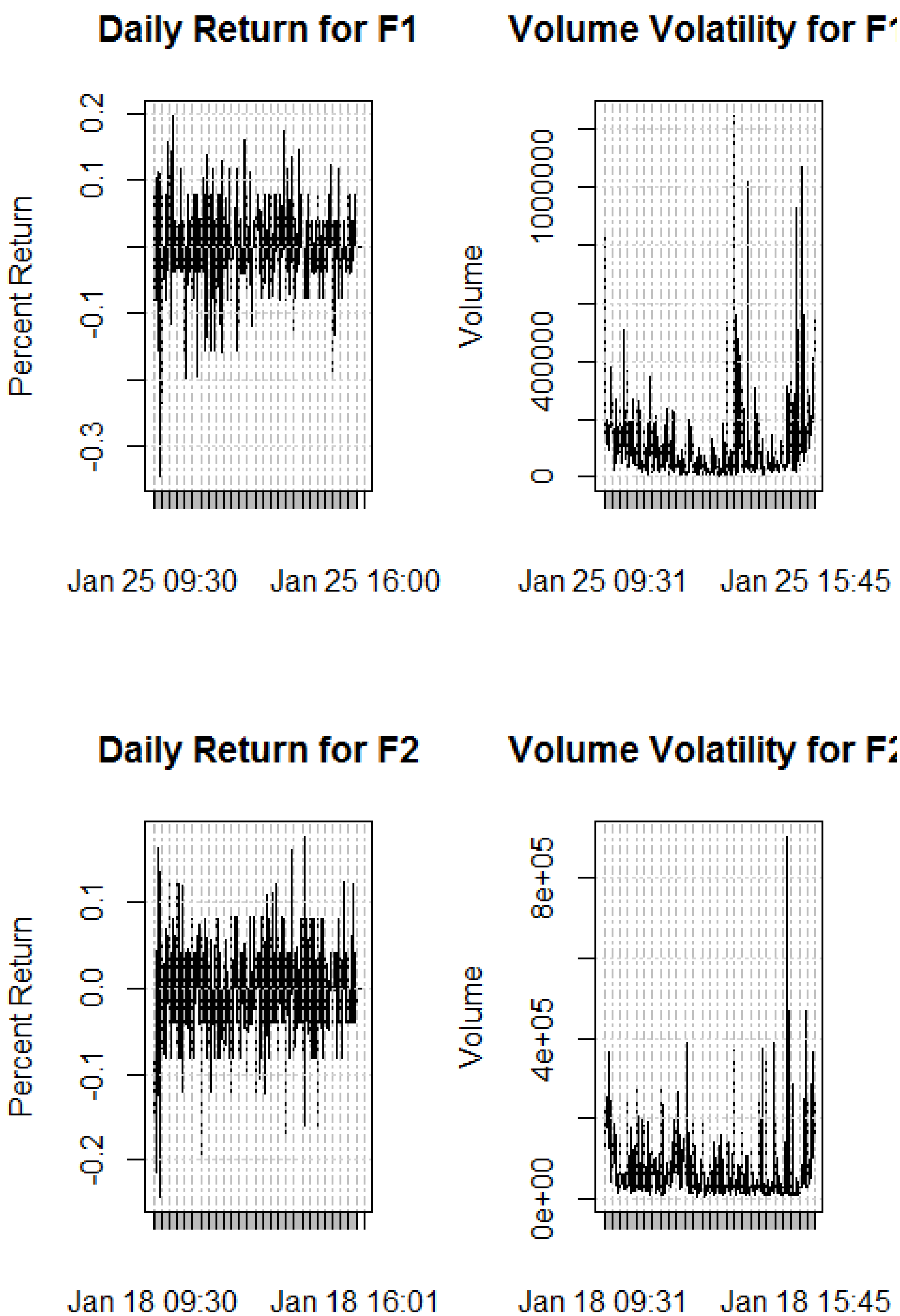

Daily Return for UTX5
Percent Return
0.1
0.0
-0.1
-0.2
-0.3
Nov 25 09:30
Nov 25 16:15

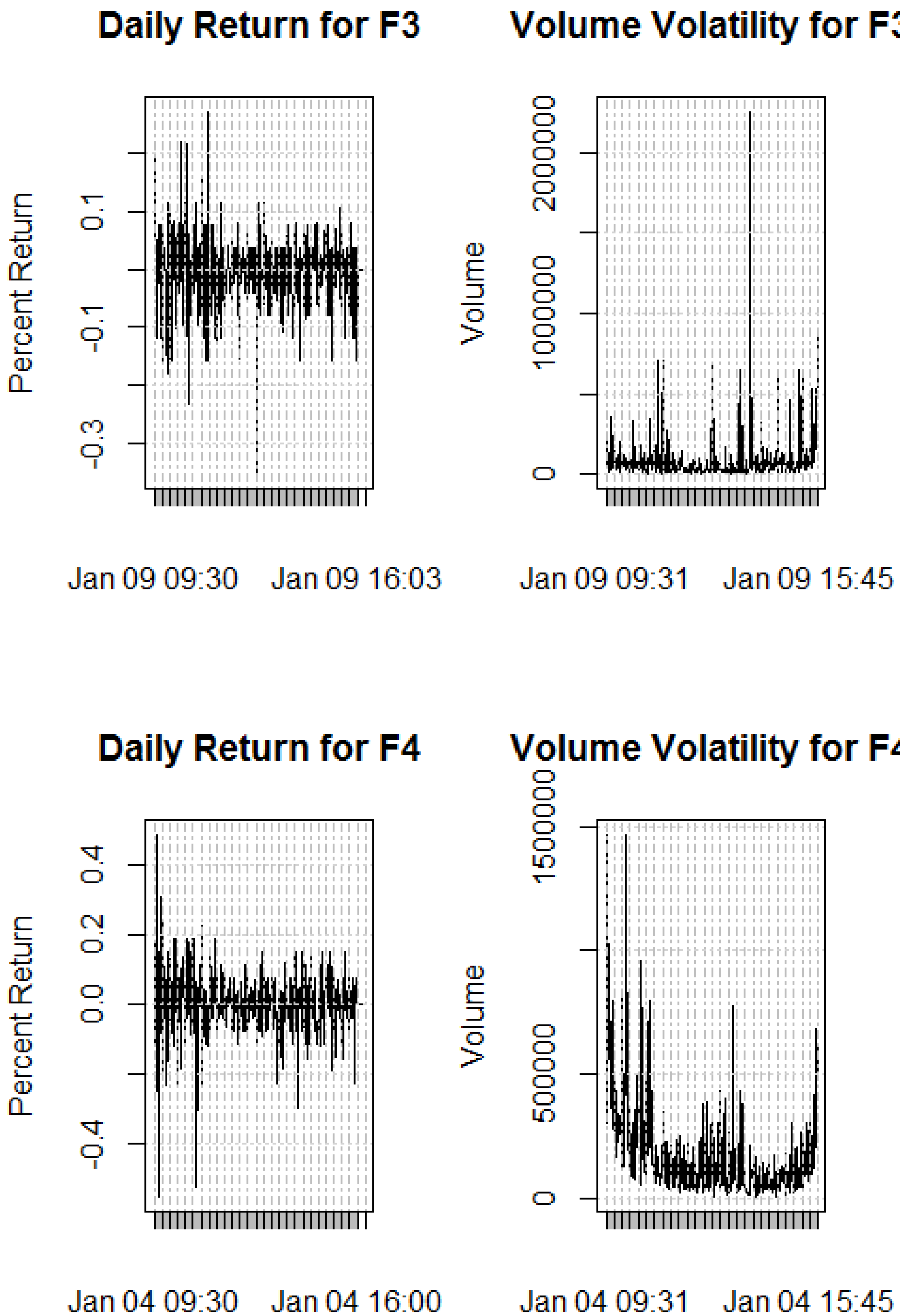

Volume Volatility for UT
Volume
0
40000
80000
Nov 25 09:31
Nov 25 13:40

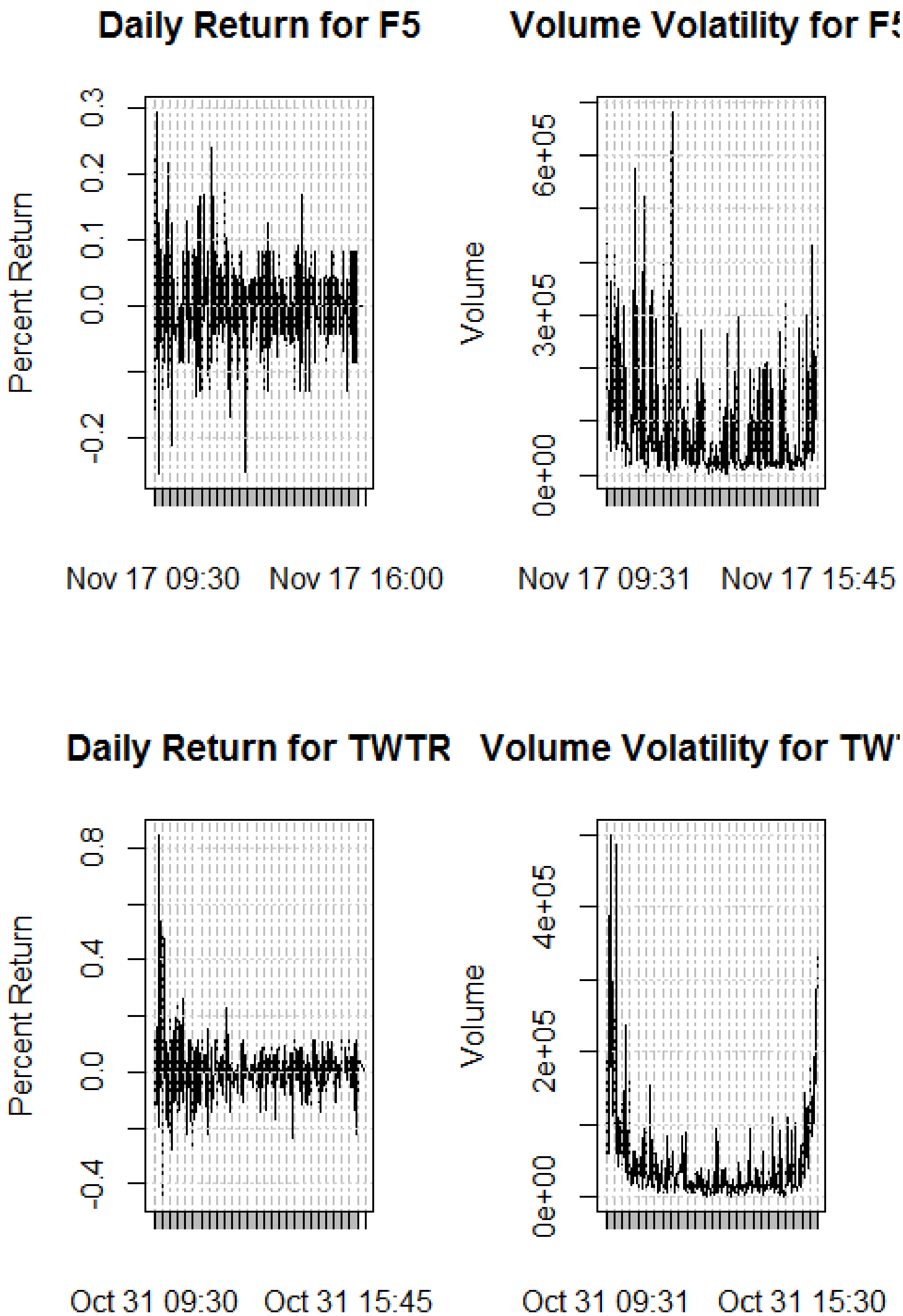

Daily Return for RXN
Percent Return
0.5
0.0
-0.5
Dec 05 09:30
Dec 05 16:02

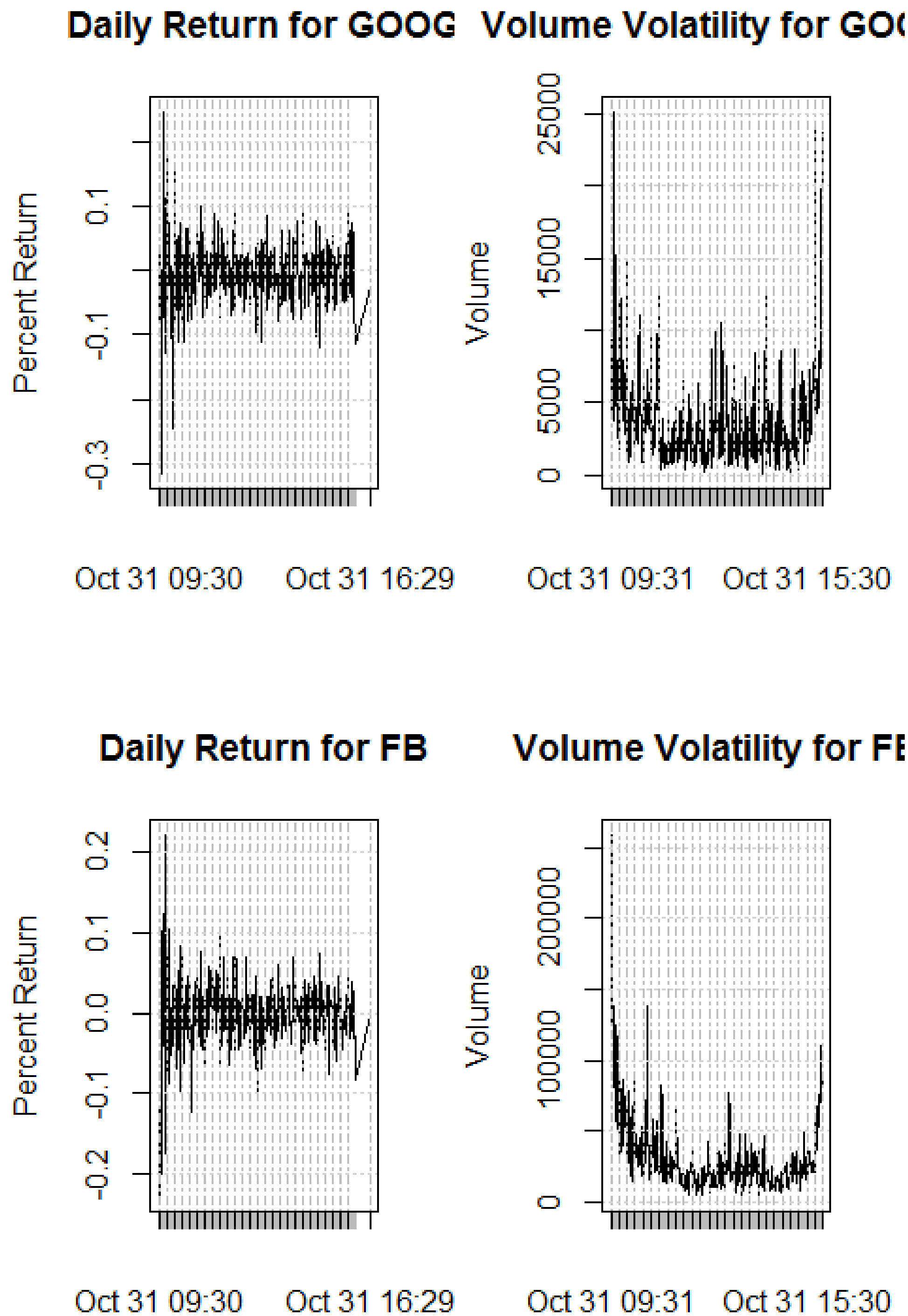

Volume Volatility for RX
Volume
0
40000
100000
Dec 05 09:31
Dec 05 15:45


Daily Return for BA1
Percent Return
0.2
0.0
-0.2
-0.4
Feb 17 09:30
Feb 17 16:00
Volume Volatility for BA
Volume
60000
40000
20000
0
Feb 17 09:31
Feb 17 15:45
Daily Return for BA2
Percent Return
0.1
0.0
-0.1
-0.2
-0.3
Dec 23 09:30
Dec 23 16:02
Volume Volatility for BA
Volume
60000
40000
20000
0
Dec 23 09:31
Dec 23 15:45


Daily Return for F1
Percent Return
0.2
0.1
-0.1
-0.3
Jan 25 09:30
Jan 25 16:00
Volume Volatility for F1
Volume
1000000
400000
0
Jan 25 09:31
Jan 25 15:45
Daily Return for F2
Percent Return
0.1
0.0
-0.1
-0.2
Jan 18 09:30
Jan 18 16:01
Volume Volatility for F2
Volume
8e+05
4e+05
0e+00
Jan 18 09:31
Jan 18 15:45


Daily Return for F3
Percent Return
0.1
-0.1
-0.3
Jan 09 09:30
Jan 09 16:03
Volume Volatility for F3
Volume
2000000
1000000
0
Jan 09 09:31
Jan 09 15:45
Daily Return for F4
Percent Return
0.4
0.2
0.0
-0.4
Jan 04 09:30
Jan 04 16:00
Volume Volatility for F4
Volume
1500000
500000
0
Jan 04 09:31
Jan 04 15:45


Daily Return for F5
Percent Return
0.3
0.2
0.1
0.0
-0.2
Nov 17 09:30
Nov 17 16:00
Volume Volatility for F5
Volume
6e+05
3e+05
0e+00
Nov 17 09:31
Nov 17 15:45
Daily Return for TWTR
Percent Return
0.8
0.4
0.0
-0.4
Oct 31 09:30
Oct 31 15:45
Volume Volatility for TW
Volume
4e+05
2e+05
0e+00
Oct 31 09:31
Oct 31 15:30


Daily Return for GOOG
Volume Volatility for GOO
Percent Return
-0.3
-0.1
0.1
Oct 31 09:30
Oct 31 16:29
Volume
0
5000
15000
25000
Oct 31 09:31
Oct 31 15:30
Daily Return for FB
Volume Volatility for FE
Percent Return
-0.2
-0.1
0.0
0.1
0.2
Oct 31 09:30
Oct 31 16:29
Volume
0
100000
200000
Oct 31 09:31
Oct 31 15:30

## STEP 3: Utilize the sentiment from Donald Trump's tweets to ultimately create an ETF.

First, we calculated the word frequencies within Donald Trump's tweets

```
trump_tweets_df <- read_csv('C:/Users/neste/Google Drive/Stevens/FE-
582(DataScience)/group_project/trump_tweets.csv')
data("stop_words")
trump_tweets_tidy <- trump_tweets_df %>%
     mutate(text = toupper(iconv(text))) %>%
     unnest_tokens(output = word, text) %>%
     anti_join(stop_words) %>%
     count(word, sort=T) %>%
     ungroup()

head(trump_tweets_tidy, 20)
```

```
## # A tibble: 20 × 2
##              word     n
##             <chr> <int>
## 1           https   379
## 2            t.co   379
## 3             amp    89
## 4          people    64
## 5           trump    52
## 6         america    46
## 7  draintheswamp     46
## 8            time    46
## 9            join    44
## 10            u.s    44
## 11        country    42
## 12        hillary    42
## 13       election    41
## 14          media    41
## 15        clinton    40
## 16           jobs    39
## 17              â    38
## 18           news    38
## 19      president    36
## 20           fake    33
```

Next, we needed to establish a positive/negative dictionary to evaluate the sentiment of each tweet. To do this, we incorporated the most frequently used words in each of Trumps tweets which we personally marked as positive or negative, as well as the Stanford NLP dictionary of positive and negative words.

```
#Read in Team 7's modified sentiment dictionary based on the word frequency
in Trump Tweets
tmp <- read_csv('C:/Users/neste/Google Drive/Stevens/FE-
582(DataScience)/group_project/trump_freq_words.csv')
```

```r
colnames(tmp) <- c("word", "sentiment")
tmp <- tmp[!is.na(tmp$sentiment), ]
nrow(tmp)
```

```
## [1] 57
```

```r
tmp_list <- lapply(split(tmp, tmp$sentiment), function(x) x$word)
tmp_negative <- tmp_list$`0`
length(tmp_negative)
```

```
## [1] 32
```

```r
tmp_positive <- tmp_list$`1`
length(tmp_positive)
```

```
## [1] 25
```

```r
positive_words <- toupper(unlist(read.delim(file = 'C:/Users/neste/Google
Drive/RWD/NLP/positive-words.txt', stringsAsFactors =
F)));names(positive_words) <-NULL
positive_words <- positive_words[-c(1:32)]
print(sprintf("The number of positive words in this vector is:
%d",length(positive_words)))
```

```
## [1] "The number of positive words in this vector is: 2006"
```

```r
positive_words <- unique(append(positive_words, toupper(tmp_positive)))
print(sprintf("The number of positive words in this vector is NOW:
%d",length(positive_words)))
```

```
## [1] "The number of positive words in this vector is NOW: 2024"
```

```r
negative_words <- toupper(unlist(read.delim(file = 'C:/Users/neste/Google
Drive/RWD/NLP/negative-words.txt', stringsAsFactors =
F)));names(negative_words) <-NULL
negative_words <- negative_words[-c(1:32)]
print(sprintf("The number of negative words in this vector is:
%d",length(negative_words)))
```

```
## [1] "The number of negative words in this vector is: 4783"
```

```r
negative_words <- unique(append(negative_words, toupper(tmp_negative)))
print(sprintf("The number of negative words in this vector is NOW:
%d",length(negative_words)))
```

```
## [1] "The number of negative words in this vector is NOW: 4805"
```

Read in the Donald Trump tweets DataFrame, clean-up the created date/time column and convert to a date variable.

```
trump_tweets_df <- read_csv("C:/Users/neste/Google
Drive/RWD/Twitter/trump/trump_tweets.csv") %>%
     separate(created, c("Date", "Time"), sep = " ", remove = T) %>%
     mutate(Date = ymd(Date))
```

Define two functions. First function will compute the sentiment score for given text. Second function will return the number of capital letters for given text.

```
get_sentiment_score <- function(a){
     #split the string by spaces to extract each individual word from the
given text
     a_split <- unlist(strsplit(toupper(iconv(a)), " "))

     #positive count will be equal to the length of the set of words
contained within the positive dictionary and our given words.
     positive_count <- length(intersect(a_split, positive_words))
     #negative count will be equal to the length of the set of words
contained within the negative dictionary and our given words.
     negative_count <- length(intersect(a_split, negative_words))

     return(positive_count / (positive_count + negative_count))
}

get_capital_letters <- function(a){
     a_split <- unlist(strsplit(iconv(a), NULL))
     return(sum(str_detect(a_split, "[A-Z]"), na.rm = T))
}
```

For each day that Donald Trump tweets, generate the average sentiment score of his tweets. If a score happens to be NA, we will impute it to be the average of all daily sentiment scores.

```
trump_sentiment_df <- trump_tweets_df %>%
     rowwise() %>%
     mutate(sentiment_score = get_sentiment_score(text)) %>%
     mutate(capital_score = get_capital_letters(text)) %>%
     ungroup() %>%
     group_by(Date) %>%
     summarise(daily_sentiment_score = mean(sentiment_score, na.rm = T),
total_capital_letters = sum(capital_score, na.rm=T), retweets_favorites =
sum(c(retweetCount, favoriteCount),na.rm=T))

#Impute NA scores to be the average daily sentiment score.
trump_sentiment_df$daily_sentiment_score[is.na(trump_sentiment_df$daily_senti
ment_score)] <- mean(trump_sentiment_df$daily_sentiment_score,na.rm=T)
#Show 15 records of results
head(trump_sentiment_df, 15)
```

```
## # A tibble: 15 × 4
##          Date daily_sentiment_score total_capital_letters
##        <date>                 <dbl>                 <int>
## 1  2016-10-17             0.3484848                   116
## 2  2016-10-18             0.6388889                   184
## 3  2016-10-19             0.3076923                   313
## 4  2016-10-20             0.5183908                   662
## 5  2016-10-21             0.4861111                   187
## 6  2016-10-22             0.6346154                   245
## 7  2016-10-23             0.0000000                    15
## 8  2016-10-25             0.4444444                   165
## 9  2016-10-26             0.3333333                    68
## 10 2016-10-27             0.6428571                   303
## 11 2016-10-28             0.4047619                   121
## 12 2016-10-29             0.6666667                   124
## 13 2016-10-30             0.6979167                   128
## 14 2016-10-31             0.5833333                    57
## 15 2016-11-02             1.0000000                    23
## # ... with 1 more variables: retweets_favorites <dbl>
```

Read in the ticker list of nasdaq and nyse.

```
ticker_list_nyse <- read_csv("C:/Users/neste/Google Drive/Stevens/FE-
582(DataScience)/group_project/nyse.csv") %>%
     select(Symbol) %>%
     unlist
names(ticker_list_nyse) <- NULL

ticker_list_nasdaq <- read_csv("C:/Users/neste/Google Drive/Stevens/FE-
582(DataScience)/group_project/nasdaq.csv") %>%
     select(Symbol) %>%
     unlist
names(ticker_list_nasdaq) <- NULL
```

Define function to read ticker information from yahoo.

```
yahoo_read <- function(url){
     require(RCurl)
     if(url.exists(url)){
          dat <- read.table(url,header=TRUE,sep=",")
          df <- dat[,c(1,5)]
          df$Date <- as.Date(as.character(df$Date))
          return(df)
     }
}
```

Define a function to read the ticker information from yahoo, calculate the return, then run a multiple linear regression against the trump sentiment data.

```
trump_ticker_fit <- function(ticker){
     cat(ticker); cat(' ')
     ticker_url <- paste0(paste0('http://real-
chart.finance.yahoo.com/table.csv?s=',ticker,'&a=07&b=24&c=2010&d=12&e=22&f=2
016&g=d&ignore=.csv'))

     ticker_df <- try(yahoo_read(ticker_url), silent = T)
     if(any(class(ticker_df) == "try-error", nrow(ticker_df) < 20,
is.null(ticker_df))){
          return(NA)
     }
     ticker_df$returns <- CalculateReturns(ts(ticker_df$Close))


     trump_vs_ticker_df <- trump_sentiment_df %>%
          inner_join(select(ticker_df, Date, returns), by = "Date")
     cor(trump_vs_ticker_df[,-1])

     fit <- try(lm(returns ~., data = trump_vs_ticker_df[,-1]), silent = T)
     if(class(fit) == "try-error"){
          return(NA)
     }
     rsq <- summary(fit)$r.squared
     return(rsq)
}
```

Run the model for each ticker in NYSE and NASDAQ, then upload all of this information into a single DataFrame.

```
stocks_to_invest <- pbsapply(c(ticker_list_nyse, ticker_list_nasdaq),
trump_ticker_fit)

stocks_to_invest_df <- data.frame(ticker = names(stocks_to_invest), rsq =
stocks_to_invest, stringsAsFactors = F)
```