

# Correlation between Financial News and Stock Market

## Final Report

Liting Hu

December 13, 2016

## 1 Introduction

Stock market is an active part of nowadays financial market. Both investors and speculators in the market would like to make better profits by analyzing market information. Financial news articles, known as one major source of market information, are widely used and analyzed by investors. In Big Data era, the amount of news articles has been increasing tremendously. In front of such a big volume of news pieces, more and more institutions rely on the high processing power of modern computers for analysis. Predictions given by support systems could assist investors to filter noises and make wiser decisions. Therefore, how to model and analyze news articles so as to make more accurate predictions becomes an interesting problem.

In this project, we will use NYT API to get articles posted in NYT website, more specifically, in Business section. And then process these data in R and judge the tendentiousness of each article. Combined with historical stock data, apply logistic regression to predict the

## 2 Get the data

### 2.1 Get NYT data

A NYT API key can be applied via <https://developer.nytimes.com/>. We use the Article Search API to get articles and access them to R by package “rtimes”. It can only search articles by page thus we need to get page number and write a loop to extract articles by page. In this report, we take “Apple” the Technology company (or “AAPL”) as an example.

```
# Get New NYT article data-----  
key = "325da26d73604830b77d8337c5afcc4a"  
q = "APPLE" # Query
```

```

fq = 'section_name:("Business") AND type_of_material:("News")' # Filtered search query
begin_date <- "20160101"
end_date <- "20161130"
sort <- "oldest"

# Search articles by key words and page
search_articles <- function(pag) {
  as <- as_search(key = key, q = q, fq = fq,
    begin_date = begin_date,
    end_date = end_date,
    sort = sort, page = pag)
  data <- as$data
  return(data)
}
# Get articles
articles <- search_articles(0)
tocountpages <- as_search(key = key, q = q, fq = fq,
  begin_date = begin_date, end_date = end_date,
  sort = sort, page = 0)
pages <- as.numeric(floor(tocountpages$meta["hits"]/10)) # Page numbers

```

Additionally, there is one problem that we can only get the first paragraph (snippet) from every article by using this package. So I added a function to get article body text by url. (<http://brooksandrew.github.io/simpleblog/articles/new-york-times-api-to-mongodb/#extracting-and-parsing-the-article-body-text>)

```

# Get article body texts by url
parseArticleBody <- function(artHTML) {
  xpath2try <- c('//div[@class="articleBody"]//p',
    '//p[@class="story-body-text story-content"]',
    '//p[@class="story-body-text"]'
  )
  for(xp in xpath2try){
    bodyi <- paste(xpathSApply(htmlParse(artHTML), xp, xmlValue), collapse='')
    if(nchar(bodyi)>0) break
  }
  return(bodyi)
}

```

## 2.2 Get stock data

In R, we can easily get stock price via package “quantmod”. What I need is stock daily return and volume so I wrote a small function to store them in a data frame. Use “AAPL” as example:

```

getstock <- function(stock.name, prd) {

```

```

stock <- getSymbols(stock.name, from = begin.date, to = end.date, auto.assign =
prdr <- periodReturn(stock, period = "daily")
prdr <- as.data.frame(prdr)
stock <- as.data.frame(stock)
date <- rownames(stock)
volume <- stock[, 5]
a <- data.frame("Date" = date, "Return" = prdr, "Volume" = volume)
return(a)
}
aapl <- getstock("AAPL")
aapl$Date <- ymd(aapl$Date)

```

	Date	daily.returns	Volume
2016-01-04	2016-01-04	2.670302e-02	67649400
2016-01-05	2016-01-05	-2.505932e-02	55791000
2016-01-06	2016-01-06	-1.956968e-02	68457400
2016-01-07	2016-01-07	-4.220457e-02	81094400
2016-01-08	2016-01-08	5.287735e-03	70798000
2016-01-11	2016-01-11	1.619224e-02	49739400
2016-01-12	2016-01-12	1.451335e-02	49154200
2016-01-13	2016-01-13	-2.571028e-02	62439600
2016-01-14	2016-01-14	2.187081e-02	63170100
2016-01-15	2016-01-15	-2.401527e-02	79010000
2016-01-19	2016-01-19	-4.838804e-03	53087700
2016-01-20	2016-01-20	1.344889e-03	72334400
2016-01-21	2016-01-21	-5.062486e-03	52161500
2016-01-22	2016-01-22	5.316713e-02	65800500
2016-01-25	2016-01-25	-1.952274e-02	51794500
2016-01-26	2016-01-26	5.530933e-03	75077000

Figure 1:

## 2.3 Other necessary information

Positive and negative words can be downloaded from Github (<https://github.com/jeffreybreen/twitter-sentiment-analysis-tutorial-201107/tree/08a269765a6b185d5fdata/opinion-lexicon-English>). Then access to R by

```

pwords <- scan('positive-words.txt', what='character')
nwords <- scan('negative-words.txt', what='character')

```

## 3 Process the data

### 3.1 Process text data

By package “tm”, we can remove punctuations and stopwords from every articles like:

```
processtext <- Corpus(VectorSource(texttoprocess))
processtext <- tm_map(processtext, removePunctuation)
processtext <- tm_map(processtext, content_transformer(tolower))
processtext <- tm_map(processtext, removeWords, stopwords("en"))
```

My original thought is to count the numbers of positive words deducting negative words as a score which shows the polarity of an article.

During the process of looking up reference, I found a package called “sentimentr” which can approximate the sentiment (polarity) of text by sentence. That is similar to what we did to count score. I calculated both of them showed in one data frame.

	Date	daily.returns	Volume	sd	ave	Score	class
1	2016-01-04	2.670302e-02	67649400	0.39322664	0.079527959	17	UP
2	2016-01-05	-2.505932e-02	55791000	0.29495015	0.179623164	23	DOWN
3	2016-01-05	-2.505932e-02	55791000	0.59222972	-0.029598657	2	DOWN
4	2016-01-06	-1.956968e-02	68457400	0.38043476	0.105462065	3	DOWN
5	2016-01-06	-1.956968e-02	68457400	0.56381894	-0.076986535	-5	DOWN
6	2016-01-06	-1.956968e-02	68457400	0.32008453	0.145188924	7	DOWN
7	2016-01-07	-4.220457e-02	81094400	0.41270131	-0.031648764	4	DOWN
8	2016-01-07	-4.220457e-02	81094400	0.52070895	-0.059401035	5	DOWN
9	2016-01-12	1.451335e-02	49154200	0.52262539	-0.048519437	-3	UP
10	2016-01-13	-2.571028e-02	62439600	0.87088356	-0.144731097	-10	DOWN
11	2016-01-25	-1.952274e-02	51794500	0.39023277	0.052005914	5	DOWN
12	2016-01-26	5.530933e-03	75077000	0.49266639	0.072718213	4	UP
13	2016-01-28	7.171891e-03	55678800	0.54003639	-0.223984035	-20	UP
14	2016-01-28	7.171891e-03	55678800	0.65618772	0.226836158	5	UP
15	2016-01-29	3.454140e-02	64416500	0.26294670	-0.085065031	-11	UP
16	2016-01-29	3.454140e-02	64416500	0.29816444	-0.077784131	-2	UP

Figure 2:

## 4 Some plots

### 4.1 Article number and word count by day

As we can see in this plot, the article numbers in weekends are much less than these in weekdays and reach its peak in Wednesday.

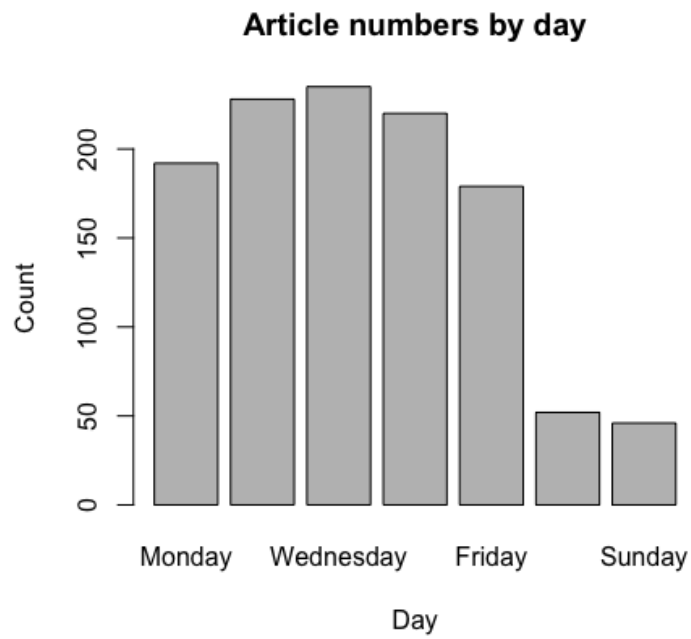


Figure 3:

As for word counts, no much difference showed between days.

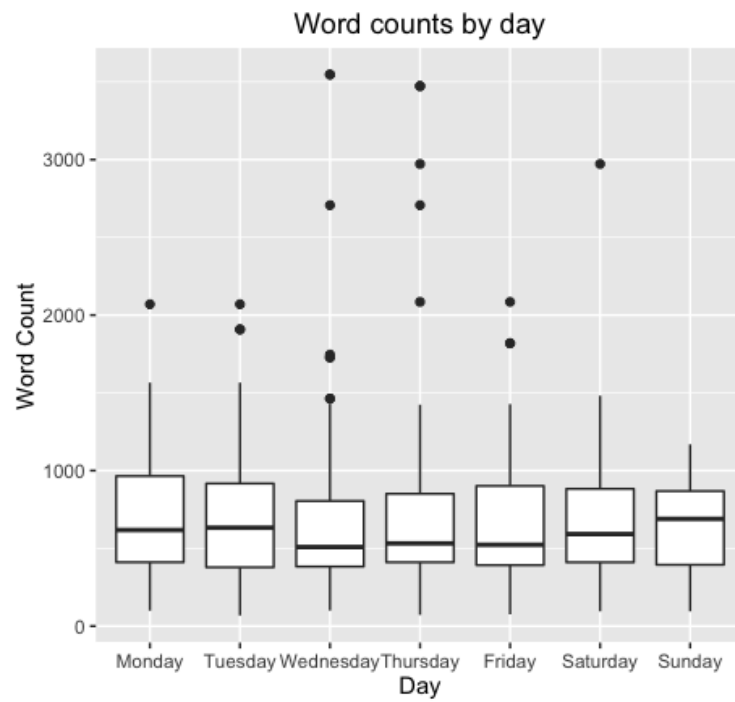


Figure 4:

## 4.2 Time series data

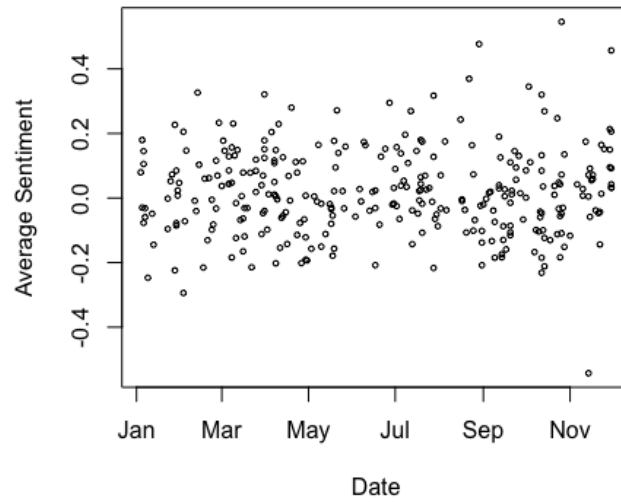


Figure 5: Sentiment Value by Date

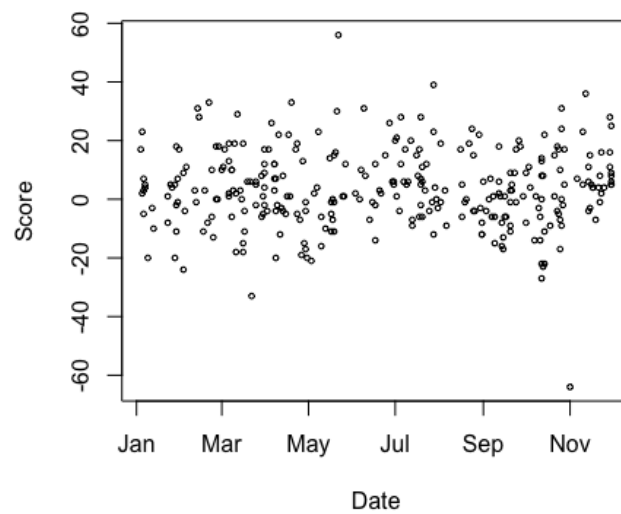


Figure 6: Sentiment Standard Deviation by Date

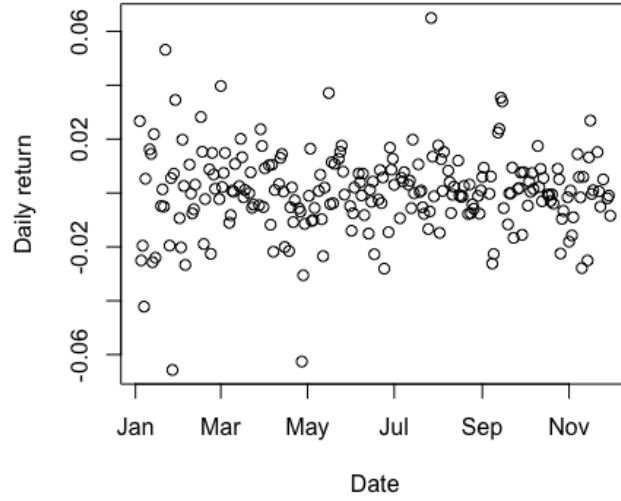


Figure 7: Stock Daily Return

Just like stock daily return, the sentiment value and the score fluctuate with no rule.

### 4.3 Two variable plots

The two plots below show the relationships between sentiment average value/standard deviation and stock return/volume.

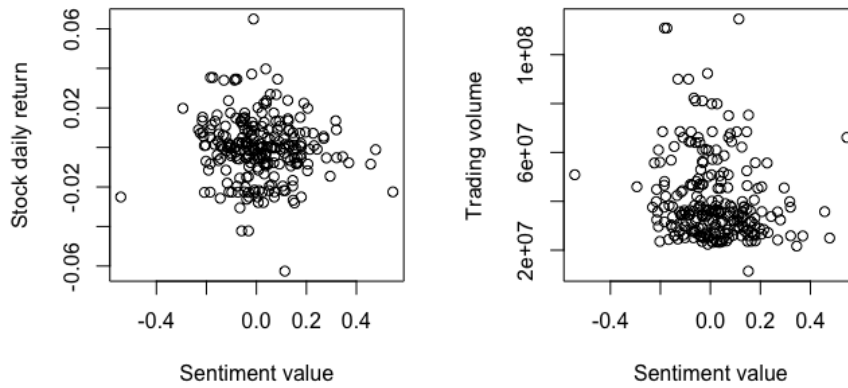


Figure 8:

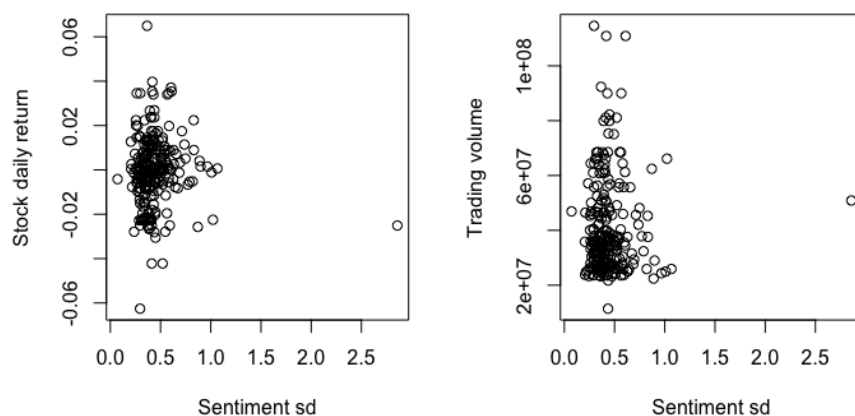


Figure 9:

And the relationships between score and stock return/volume are drew below.

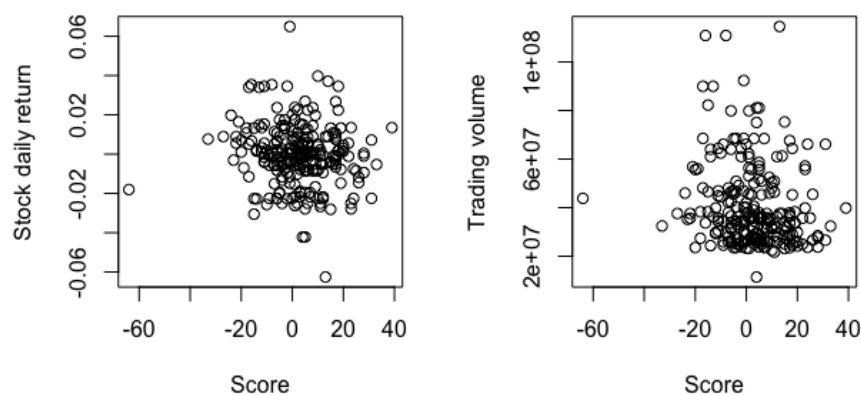


Figure 10:

From these plot, we can not see obvious correlations between our sentiment/score and stock performance. So we need to do some regression analysis to judge whether our method is efficient.

## 5 Logistic Regression

To use logistic regression, we divide all data into two group: 2/3 training set and 1/3 test set.

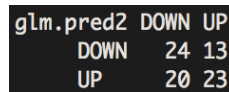


One way is to predict by score.

```
# Regression by score
glm.fit2 <- glm(class~Score, data = TrainingSet, family=binomial)
summary(glm.fit2)
glm.probs2 <- predict(glm.fit2, TestSet, type ="response")

glm.pred2=rep("DOWN", nrow(TestSet))
glm.pred2[glm.probs2 > .5]="UP"

table(glm.pred2, TestSet$class)
mean(glm.pred2 == TestSet$class)
# 0.5875
```



glm.pred2	DOWN	UP
DOWN	24	13
UP	20	23

Figure 11:

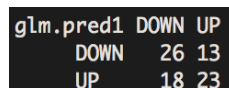
The accuracy of prediction is 0.5875.

Another way is to predict by sentiment value and its standard deviation.

```
# Regression by sentiment value
glm.fit1 <- glm(class~sd + ave, data = TrainingSet, family=binomial)
summary(glm.fit1)
glm.probs1 <- predict(glm.fit1, TestSet, type ="response")

glm.pred1=rep("DOWN", nrow(TestSet))
glm.pred1[glm.probs1 > .5]="UP"

table(glm.pred1, TestSet$class)
mean(glm.pred1 == TestSet$class)
# 0.6125
```



glm.pred1	DOWN	UP
DOWN	26	13
UP	18	23

Figure 12:

The accuracy of predictions of Logistic Regression are 0.5875 and 0.6125 which are acceptable.