Exploratory Data Analysis

2024-03-06

Introduction

Market sentiments refer to how investor view a particular company or industry. People market sentiment can be influencing pattern of trading activity.In fact, there is evidence for a causal link from sentiments to stock price returns, volatility and volume (Checkley et al., 2017). The sentiment in stock market news can be positive or negative, influenced by whether the reported information is beneficial or detrimental to the stock market (Loan et al., 2023). It was discovered that news significantly influences the purchasing patterns in the S&P NIFTY index futures, with the most pronounced effect occurring within a five-minute lag period (Ritu et al., 2019). However, the use of market sentiment to predict stock prices presents challenges, underscoring the importance of accounting for confounding factors. For example, during Covid, the consumer sentiment for sure dropped, but the stock index, in contrast, rises. And this counter-intuition market performance was actually because the Federal Reserve stepped in to support stock market.

Research goal and dataset introduction

In the pursuit of assessing the impact of news sentimenton the day-to-day fluctuations of the S&P 500 index, this study leverages two primary data sources, encompassing news text and stock data. A comprehensive dataset spanning a four-year period, from 2016 to 2019, is curated to facilitate effective model training while managing computational costs efficiently.

The news data is obtained from the New York Times Developer Archive API, selected for its reputable and authoritative standing as a leading news platform in the United States. The dataset comprises a total of 242,068 entries of news articles, exhibiting no missing values and featuring an subheading for each article.

The S&P 500 index data is acquired through Yahoo Finance's API for its convenience and reliability. The dataset spans a total of 1,005 trading days, and includes information such as opening, closing, highest, and lowest prices for each trading day.

1. market news data

1.1 Data collection

To begin with, we want to retrieve the market news data using the New York Times Developer Archive API. We choose the news data from New York times because we believe its high market penetration rate makes it the most representative for market sentiment influencer.

Before we start, we import the library we want to use for the project. To be more specific, 'htttr' is used to make web requeest and 'jsonlite' is used to parse the JSON responses returned by web service and convert into R data structurees for analysis and visualization.

```
options(repos = c(CRAN = "https://cloud.r-project.org/"))
options(warn = -1)
library(httr)
library(jsonlite)
```

```
#create a new directory named news_data in the current working directory
dir.create("news_data", showWarnings = FALSE)

# add the API keys used to retrieve the data
APIKEY <- "OOab5JPxZfUMFSqudqigXeokcCdAqhtk"

# iterate over the years 2016 to 2019 and for each year, iterate over all months (1 to 12)
for(year in 2016:2019) {
  for(month in 1:12) {
    #generate a filename of the format archive_YEAR_MONTH.json
    output_file <- sprintf("news_data/archive_%d_%02d.json", year, month)
    #A URL to make the GET request to the New York Times Archive API, incorporating the year, month, and the API
key into the request URL.
    url <- sprintf("https://api.nytimes.com/svc/archive/v1/%d/%d.json?api-key=%s", year, month, APIKEY)

    # Make the GET request and save the response
    response <- GET(url)

    # Write the content of the response to the file
    writeBin(content(response, "raw"), output_file)
  }
}
```

By the end of the script, the news_data directory will contain a JSON file for each month of each year from 2016 to 2019, each file containing news data from that particular month as provided by The New York Times Archive API.

Next, we will be processing JSON files containing news data, the aim is to extract specific peices of information from each article and compile into a single, consolidated dataframe called 'marketnews_data_raw'.

```
library(jsonlite)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Our data acquisition involves a three steps process: 1. initializing an empty data frame with the specified columns.

2. Defines a function called json_to_pdf to load JSON data from a file, select specific columns, and append this data to an existing data frame.
3. Then, we iterates through the specified years and months, applying json_to_pdf function to accumulate data into the data frame.

```
# Define columns of interest
columns <- c('abstract', 'web_url', 'snippet', 'pub_date', 'document_type', 'news_desk', 'type_of_material', 'wor
d_count', 'headline.main')

# Initialize an empty data frame with specified columns
# Adjust column names to valid R variable names where necessary
marketnews_data_raw <- setNames(data.frame(matrix(ncol = length(columns), nrow = 0)),
               gsub("headline.main", "headline_main", columns))
```

```
json_to_df <- function(json_file_path, dataframe) {
  # This function reads a JSON file, extracts the 'docs' section,
  # filters for specific columns, and appends the data to a given dataframe.

  # Args:
  #   json_file_path: The file path to the JSON file.
  #   dataframe: An existing dataframe to which the new data will be appended.

  # Returns:
  #   A dataframe with the combined original and new data from the JSON file.

  # Read and flatten the JSON file, extracting the 'docs' section
  content <- fromJSON(json_file_path, flatten = TRUE)$response$docs

  # Convert the extracted content to a dataframe
  df <- as.data.frame(content)

  # Filter for the existence of expected columns, replacing 'headline.main'
  # with 'headline_main' in the column names
  df <- df[, colnames(df) %in% gsub("headline.main", "headline_main", columns), drop = FALSE]

  # Combine the new data with the existing dataframe
  # If there are new columns, fill the gaps with NA
  dataframe <- rbind(dataframe, df, fill = TRUE)

  # Return the combined dataframe
  return(dataframe)
}
```

```
# Iterate over years and months, reading and appending data
for (year in 2016:2019) {
  for (month in 1:12) {
    filename <- sprintf("news_data/archive_%d_%02d.json", year, month)
    if(file.exists(filename)) { # Check if the file exists to avoid errors
      marketnews_data_raw <- json_to_df(filename, marketnews_data_raw)
    }
  }
}

# Convert factors to characters, if necessary
marketnews_data_raw[] <- lapply(marketnews_data_raw, as.character)
```

```
#we want to save the marketnews_data locally
write.csv(marketnews_data_raw, file = "myDataFrame.csv", row.names = FALSE)
```

```
marketnews_data_raw<- read.csv("myDataFrame.csv")
```

By the end of the script, we get our raw dataframe: marketnews_data_raw, containing a comprehensive collection of selected data fields from New York Times articles published from 2016 to 2019.

```
options(width = 100)  # Adjust the width of the output to prevent wrapping
# head(marketnews_data_raw)
knitr::kable(marketnews_data_raw[1:6,], format="markdown")
```

abstract	web_url	snippet	pub_date	document_type	news_desk	type
Bill Cosby came close to escaping sexual assault charges. How many others do?	https://www.nytimes.com/2016/01/01/opinion/no-more-statutes-of-limitations-for-rape.html (https://www.nytimes.com/2016/01/01/opinion/no-more-statutes-of-limitations-for-rape.html)	Bill Cosby came close to escaping sexual assault charges. How many others do?	2016-01-01T00:25:44+0000	article	OpEd	Op-
A court document made public on Thursday suggests that defense lawyers will argue that Mr. Gray was already injured when he was loaded into a Baltimore police van.	https://www.nytimes.com/2016/01/01/us/freddie-gray-complained-of-bad-back-before-arrest-filing-says.html (https://www.nytimes.com/2016/01/01/us/freddie-gray-complained-of-bad-back-before-arrest-filing-says.html)	A court document made public on Thursday suggests that defense lawyers will argue that Mr. Gray was already injured when he was loaded into a Baltimore police van.	2016-01-01T00:32:23+0000	article	National	New
The average tax rate for America's top 400 income earners rose in 2013, but tax analysts said the tax structure still overwhelmingly favors the very well-off.	https://www.nytimes.com/2016/01/01/business/economy/justice-in-taxes-most-likely-short-lived.html (https://www.nytimes.com/2016/01/01/business/economy/justice-in-taxes-most-likely-short-lived.html)	The average tax rate for America's top 400 income earners rose in 2013, but tax analysts said the tax structure still overwhelmingly favors the very well-off.	2016-01-01T00:45:44+0000	article	Business	New
Led by quarterback Deshaun Watson, the Tigers defeated fourth-seeded Oklahoma in the first College Football Playoff semifinal.	https://www.nytimes.com/2016/01/01/sports/ncaafootball/clemson-beats-oklahoma-in-orange-bowl-college-football-playoff-semifinal.html (https://www.nytimes.com/2016/01/01/sports/ncaafootball/clemson-beats-oklahoma-in-orange-bowl-college-football-playoff-semifinal.html)	Led by quarterback Deshaun Watson, the Tigers defeated fourth-seeded Oklahoma in the first College Football Playoff semifinal.	2016-01-01T00:53:02+0000	article	Sports	New
A federal judge has warned that prosecutors may be going too far when they ask witnesses to keep quiet about receiving a subpoena.	https://www.nytimes.com/2016/01/01/nyregion/prosecutorssecrecy-orders-on-subpoenas-stir-constitutional-questions.html (https://www.nytimes.com/2016/01/01/nyregion/prosecutorssecrecy-orders-on-subpoenas-stir-constitutional-questions.html)	A federal judge has warned that prosecutors may be going too far when they ask witnesses to keep quiet about receiving a subpoena.	2016-01-01T00:59:43+0000	article	Metro	New

abstract	web_url	snippet	pub_date	document_type	news_desk	type
William Ackman's hedge fund said in a regulatory filing that its stake was now 8.5 percent, down from 9.9 percent, of the drug company that has been drawing fire on several fronts.	https://www.nytimes.com/2016/01/01/business/dealbook/ackman-sheds-some-valeant-shares.html (https://www.nytimes.com/2016/01/01/business/dealbook/ackman-sheds-some-valeant-shares.html)	William Ackman's hedge fund said in a regulatory filing that its stake was now 8.5 percent, down from 9.9 percent, of the drug company that has been drawing fire on several fronts.	2016-01-01T01:06:43+0000	article	Business	New

Examining our dataset, Marketnews_data_raw contain 242068 rows and 8 columns. It contains all the market news information on yahoo news each day from 2016-2019. To be more specific, the columns names are: snippet/abstract: the subeheading of news pub_date: the date that the news is publicated document_type: type of news article news_desk: section of news word_count: number of words in a news article headline: the headline of news

1.2 data cleaning

On our second step, we will be checking whether there are null or missing values in our targeted columns which are pub_date and snippet.

```
# Check how many missing values are in 'pub_date' and 'snippet'
missing_pub_dates <- sum(is.na(marketnews_data_raw$pub_date))
missing_snippets <- sum(is.na(marketnews_data_raw$snippet))

# Print out the number of missing values for inspection
print(paste("Missing 'pub_date':", missing_pub_dates))
```

```
## [1] "Missing 'pub_date': 0"
```

```
print(paste("Missing 'snippet':", missing_snippets))
```

```
## [1] "Missing 'snippet': 0"
```

Fortunately, our targted columnns do not contain any missing values!

1.3 data preprocessing

And we will be performing some natural language processing task for performing machine learning in future work.

First, we want to install and import the package for natural language processing. “tm” is versatile for text mining, including its ability in managing text documents, manipulating document.

```
install.packages("tm")
```

```
## The following package(s) will be installed:
## - tm [0.7-13]
## These packages will be installed into "~/Desktop/Spring 2024/QTM_302/Project/renv/library/R-4.3/aarch64-apple-darwin20".
##
## # Installing packages --------------------------------------------------------
## - Installing tm ...                            OK [linked from cache]
## Successfully installed 1 package in 3.6 milliseconds.
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:httr':
##
##     content
```

Next, we define a function called tokenize_and_stem_optimized that does 4 things 1. lower case all the abstract 2. removing all the stop words 3. only considering the alphabetic characters 4. stematize the document

```
tokenize_and_stem_optimized <- function(text_vector) {
  # This function tokenizes and stems a given vector of text strings.
# It converts the text to lowercase, removes punctuation, removes English stopwords,
# and applies stemming to reduce words to their root forms.
#
# Args:
#   text_vector: A character vector where each element is a text string that will be tokenized and stemmed.
#
# Returns:
#   A character vector of the same length as text_vector, where each element
#   is the processed version of the corresponding element in text_vector,
#   with text converted to lowercase, punctuation removed, stopwords removed,
#   and words stemmed.


  # Ensure the text_vector is a character vector
  text_vector <- as.character(text_vector)

  # Create a corpus from the entire vector of text (batch processing)
  docs <- Corpus(VectorSource(text_vector))

  # Preprocess: Convert to lowercase, remove punctuation, remove stopwords, stem the document
  # These steps are now applied to the entire corpus in a batch, which is more efficient
  docs <- tm_map(docs, content_transformer(tolower))
  docs <- tm_map(docs, removePunctuation)
  docs <- tm_map(docs, removeWords, stopwords("english"))
  docs <- tm_map(docs, stemDocument)

  # Convert back to text
  # Use lapply and unlist to efficiently handle list output and flatten it to a character vector
  text_stemmed <- sapply(docs, as.character)

  return(text_stemmed)
}
```

finally, we apply our function to the snippet columnn of the data and we create a new table called marketnews_data_preprossessed that contain 2 columns: the publication date and the preprossessed(lemmatized, lowercased, etc) of the original data

```
# Apply the optimized function to the entire columns at once
marketnews_data_raw$snippet <- tokenize_and_stem_optimized(marketnews_data_raw$snippet)

marketnews_data_preprossessed <- data.frame(pub_date = marketnews_data_raw$pub_date, snippet = marketnews_data_ra
w$snippet)
```

```
#we want to save the proprossed date locally
write.csv(marketnews_data_preprossessed, file = "marketnews_data_preprossessed.csv", row.names = FALSE)
```

1.4 Data analysis and data visulization

Now we have our finalized market news dataframe, we want to perform some basic statistical metrics and data visualization.

```
marketnews_data_preprossessed = read.csv("marketnews_data_preprossessed.csv")
```

```
library(dplyr)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

first, we aggregate each day and count the total number of snippets

```
# Convert pub_date to Date type and extract date part only
marketnews_data_preprossessed$pub_date <- as.Date(ymd_hms(marketnews_data_preprossessed$pub_date))

#Group by publication date and count snippets
df_grouped <- marketnews_data_preprossessed %>%
  group_by(pub_date) %>%
  summarise(count = n())
```

second, we get the central tendency measures of the snippet on each day

```
#first we want to filter out the day with max count to avoid outlier
df_grouped <- df_grouped[-1, ]
df_grouped<- df_grouped%>%
  filter(df_grouped$count!= max(df_grouped$count))

#some central tendency measures
mean_snippets <- mean(df_grouped$count)
median_snippets <- median(df_grouped$count)
var_snippets<- var(df_grouped$count)
print(mean_snippets)
```

```
## [1] 201.9392
```

```
print(median_snippets)
```

```
## [1] 222
```

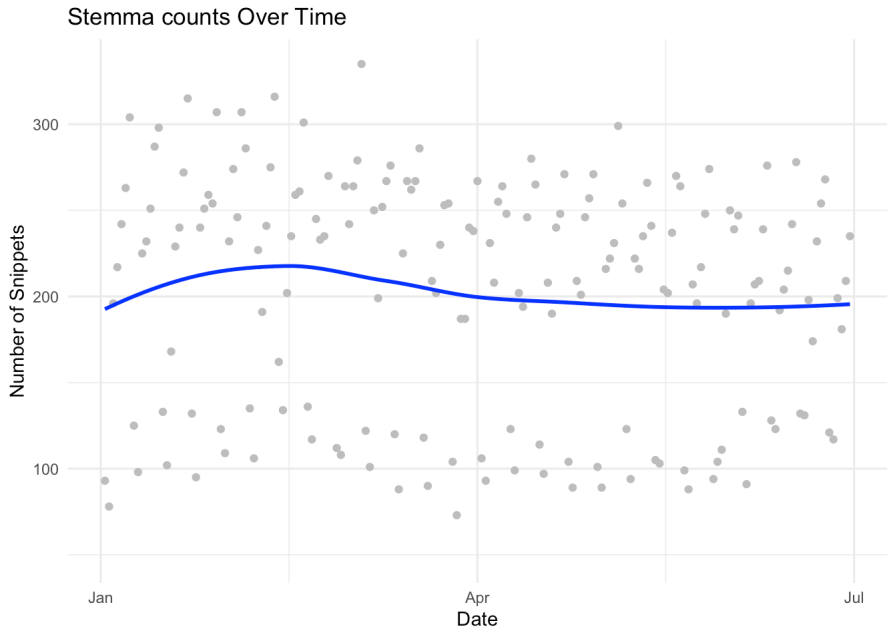
```
print(var_snippets)
```

```
## [1] 4608.924
```

On average there are 165 stemmas each day from 2016-2019, the median of the stemma is 175. The proximaty of these two numbers suggest there are not too many outliers. The snippets have a high variance, which means there are huge deviations from the means.

```
# Plotting with ggplot2
ggplot(df_grouped, aes(x = pub_date, y = count)) +
  geom_point(color="grey") +
  geom_smooth(method = "loess", color = "blue", se = FALSE)+
  labs(title = "Stemma counts Over Time",
       x = "Date",
       y = "Number of Snippets") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



As we can see from the graph, the number of lemmas decline over time but on average concentrate between 150 -210. Thus, we can infer the information carried through the news snippets remains in a roughly constant level. Hence, we are guaranteed some disruptive information explosion did not appear in our interested period.

```
library(dplyr)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

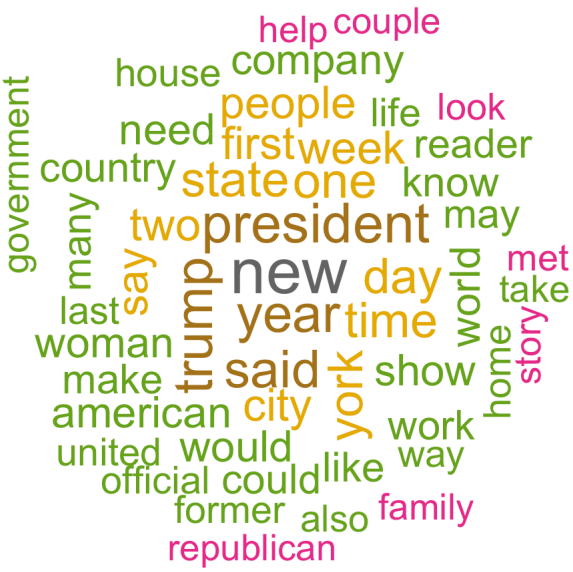
```
library(tm)  # for text manipulation

tfidf <- read.csv('tfidf_snippet.csv')
# Exclude the 'pub_date' column assuming it's the first column
tfidf_corrected <- tfidf %>% select(-pub_date)

# Calculate the average TF-IDF scores for each term
average_tfidf <- colMeans(tfidf_corrected, na.rm = TRUE)

# Get the top 50 words based on average TF-IDF scores
top_words <- sort(average_tfidf, decreasing = TRUE)[1:50]

# Generate the word cloud
wordcloud(names(top_words), top_words, max.words = 50, scale = c(3, 0.5),
          colors = brewer.pal(8, "Dark2"), random.order = FALSE)
```



We also applied the word cloud graph to show interesting words appearance vividly. Clearly, words like “President” and “Trump” cover a large area in our word cloud, indicating that the President Trump was gaining a lot of public attention.

2. yahoo news stock data

In this section, we retrieve S&P 500 stock prices and stock volatility using the Yahoo Finance API for its convenience and reliability. The dataset spans a total of 1,005 trading days, and includes information such as opening, closing, highest, and lowest prices for each trading day. A feature of daily percent change is generated as the target for prediction. Over the seven-year period from 2016 to 2023, the S&P 500 index increased by 169.05%, displaying constant fluctuations. This characteristic makes the study particularly intriguing as it delves into the role of news sentiments in influencing these fluctuations.

2.1 Stock data import

The package “quantmod” is applied to organize and present the time series data, while the package “dplyr” is utilized in the purpose of cleaning and manipulating the datasets.

```
install.packages("quantmod")
```

```
## The following package(s) will be installed:
## - quantmod [0.4.26]
## These packages will be installed into "~/Desktop/Spring 2024/QTM_302/Project/renv/library/R-4.3/aarch64-apple-darwin20".
##
## # Installing packages --------------------------------------------------------
## - Installing quantmod ...                      OK [linked from cache]
## Successfully installed 1 package in 2.9 milliseconds.
```

```
install.packages("dplyr")
```

```
## The following package(s) will be installed:
## - dplyr [1.1.4]
## These packages will be installed into "~/Desktop/Spring 2024/QTM_302/Project/renv/library/R-4.3/aarch64-apple-darwin20".
##
## # Installing packages --------------------------------------------------------
## - Installing dplyr ...                         OK [linked from cache]
## Successfully installed 1 package in 2.9 milliseconds.
```

```
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # 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.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method           from
##   as.zoo.data.frame zoo
```

```
library(dplyr)
```

In the following snippet downloads the data from Yahoo Finance API. Given the popularity of research projects related to the stock market, R has a built-in access to Yahoo Finance API, hence enabling us to easily import the stock market data of any particular stock.

```
###download the stock raw data###

# Specify stock symbol and date range

stock <- "^GSPC"
start <- as.Date("2016-01-01")
end <- as.Date("2019-12-31")

# Download stock data
df2 <- getSymbols(stock, src = "yahoo", from = start, to = end, auto.assign = FALSE)
```

We intentionally selected the time period from “2016-01-01” to “2023-12-31” because we believe the impact of Covid is only emphasized through the larger scope of observations.

2.2 raw stock data analysis

In this section we exhibit our data and demonstrate several key features. Thanks to the well-organized stock market, we enjoy the convenience that there is no need to further organize the dataset other than collecting our interested pieces.

```
head(df2)
```

```
##            GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted
## 2016-01-04   2038.20   2038.20  1989.68    2012.66  4304880000       2012.66
## 2016-01-05   2013.78   2021.94  2004.17    2016.71  3706620000       2016.71
## 2016-01-06   2011.71   2011.71  1979.05    1990.26  4336660000       1990.26
## 2016-01-07   1985.32   1985.32  1938.83    1943.09  5076590000       1943.09
## 2016-01-08   1945.97   1960.40  1918.46    1922.03  4664940000       1922.03
## 2016-01-11   1926.12   1935.65  1901.10    1923.67  4607290000       1923.67
```

```
summary(df2)
```

```
##      Index              GSPC.Open      GSPC.High       GSPC.Low      GSPC.Close
##  Min.   :2016-01-04   Min.   :1833   Min.   :1847   Min.   :1810   Min.   :1829
##  1st Qu.:2016-12-30   1st Qu.:2268   1st Qu.:2272   1st Qu.:2259   1st Qu.:2265
##  Median :2017-12-29   Median :2603   Median :2628   Median :2586   Median :2602
##  Mean   :2017-12-30   Mean   :2550   Mean   :2560   Mean   :2538   Mean   :2550
##  3rd Qu.:2018-12-31   3rd Qu.:2818   3rd Qu.:2830   3rd Qu.:2803   3rd Qu.:2816
##  Max.   :2019-12-30   Max.   :3247   Max.   :3248   Max.   :3234   Max.   :3240
##   GSPC.Volume        GSPC.Adjusted
##  Min.   :1.297e+09   Min.   :1829
##  1st Qu.:3.242e+09   1st Qu.:2265
##  Median :3.525e+09   Median :2602
##  Mean   :3.630e+09   Mean   :2550
##  3rd Qu.:3.883e+09   3rd Qu.:2816
##  Max.   :7.658e+09   Max.   :3240
```

```
df2$Change <- (df2$GSPC.Close - df2$GSPC.Open) / df2$GSPC.Open * 100

# Create the new dataframe with selected columns
df_stock <- df2[, c('GSPC.Open', 'GSPC.Close', 'Change')]

# Print the new dataframe
print(df_stock)
```

```
##            GSPC.Open GSPC.Close      Change
## 2016-01-04   2038.20    2012.66 -1.25306239
## 2016-01-05   2013.78    2016.71  0.14549413
## 2016-01-06   2011.71    1990.26 -1.06625466
## 2016-01-07   1985.32    1943.09 -2.12711208
## 2016-01-08   1945.97    1922.03 -1.23023180
## 2016-01-11   1926.12    1923.67 -0.12719619
## 2016-01-12   1927.83    1938.68  0.56281404
## 2016-01-13   1940.34    1890.28 -2.57995699
## 2016-01-14   1891.68    1921.84  1.59434530
## 2016-01-15   1916.68    1880.33 -1.89651359
##        ...
## 2019-12-16   3183.63    3191.45  0.24563372
## 2019-12-17   3195.40    3192.52 -0.09012590
## 2019-12-18   3195.21    3191.14 -0.12738031
## 2019-12-19   3192.32    3205.37  0.40879513
## 2019-12-20   3223.33    3221.22 -0.06546358
## 2019-12-23   3226.05    3224.01 -0.06323644
## 2019-12-24   3225.45    3223.38 -0.06417921
## 2019-12-26   3227.20    3239.91  0.39383866
## 2019-12-27   3247.23    3240.02 -0.22203419
## 2019-12-30   3240.09    3221.29 -0.58023229
```

2.3 stock data visualization

```
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:httr':
##
##     config
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plot(df2$GSPC.Open, type = 'l', col = 'blue',
     main = 'GSPC Open vs Close', xlab = 'Index', ylab = 'Price')
```


```
lines(df2$GSPC.Close, type = 'l', col = 'red')
legend('topright', legend = c('GSPC.Open', 'GSPC.Close'),
       col = c('blue', 'red'), lty = 1)
```

In this figure, we demonstrate how S&P 500 stock price has been changing across our interested period. The comparison between the Open price and the Close price is not significant as the volatility in one single day is largely trivialized by the changes in longer term.

```
df_GSPC <- data.frame(Date=index(df2),coredata(df2))
df_GSPC_100 <- tail(df_GSPC, 100)

fig <- df_GSPC_100 %>% plot_ly(x = ~Date, type="candlestick",
                      open = ~GSPC.Open, close = ~GSPC.Close,
                      high = ~GSPC.High, low = ~GSPC.Low)
fig <- fig %>% add_lines(x = ~Date, y = ~GSPC.Open, line = list(color = 'black', width = 0.75), inherit = F)
fig <- fig %>% layout(showlegend = FALSE)

fig
```

In this graph, we provide a more accurate depiction of the change of stock market data through a standard candlestick graph. However, in order to be more detailed, we are limited to exhibit the lastest 100 days, otherwise the graph will explode and become less informative. Such excerpt from the whole time line of stock market data demonstrates how we can observe the daily volitility of stock market price. Hence, we can manage to predict such variation through collected news datas.

3. future works.

```
Currently, we have two dataframe: marketnews_data
preprossed that contains stammatized words on each from and stock
data that contain the price change each day. For the next step, we
plan on establishing a correlationship between these two data set.
Here are a fews things that we plan to do in the future
```

- we will apply sentimental analysis(tools in R that give a sentiment score stemmas on a given day. The score varies from positive(happy, good, strong, and etc) to negative (bullish, bad, sad and etc)
- we will try to apply machine learning to train the daily sentimental scoree and daily price from a specific period of time and use that to predict stock price. A strong accuracy score can indicate strong predictive power in sentimental score on stock pricee
- we also want to control other confounders that can potentially influence stock price trend such as interest rate, market event and etc.

4. Bibliography

[1] Checkley, M., Higón, D. A., & Alles, C. (2017). The hasty wisdom of the mob: How market sentiment predicts stock market behavior. Expert Systems with Applications, 77, 256-263. https://doi.org/ (https://doi.org/) 10.1016/j.eswa.2017.01.029

[2] Loan, C. T., et al. (2023). Predicting stock market indicators through analysis of global news sentiment. Emerald Insight. [Online]. Available: https://www.emerald.com/insight/content/ (https://www.emerald.com/insight/content/) doi/10.1108/AJB-08-2022-0124/full/html.

[3]Ritu Yadav, A. Vinay Kumar, Ashwani Kumar, News-based supervised sentiment analysis for prediction of futures buying behaviour, IIMB Management Review, Volume 31, Issue 2, 2019, Pages 157- 166, ISSN 0970-3896. https://doi.org/10.1016/ (https://doi.org/10.1016/) j.iimb.2019.03.006.

data citation New York Times Company. (2016-2019). Stock price data. New York Times Developer Network. https://developer.nytimes.com/ (https://developer.nytimes.com/)

Yahoo! Inc. (2016-2019). S&P 500 index data. Yahoo Finance API. https://finance.yahoo.com/ (https://finance.yahoo.com/)