597_Stock_Analysis

Zhuofan Dong and Guangjian Li

4/24/2022

Part0: Pre-setting

Broad goal: We hope to make investment decisions during the pandemic, because this is the time that the market is under great uncertainty. Experienced investors will look for the opportunities to make profit from such uncertainty.

Datasets: 1. SP500.CSV: first 30 stocks from benchmark s&p 500 ranked by market capitalization 2. stock data from yahoo finance package in r 3. Dataset of news headlines from Kaggle.

```
setwd("C:\\Users\\Primo\\OneDrive\\Desktop\\597")
```

Import packages

```
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
```

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

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

```
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

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

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:PerformanceAnalytics':
##
##     kurtosis, skewness
```

```
library(quadprog)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
```

```
library(dplyr)
library(tidyr)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(stringr)
library(lubridate)
```

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

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

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

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

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

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

```
library(tidytext)
library(wordcloud)
```

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

```
##
## Attaching package: 'wordcloud'
```

```
## The following object is masked from 'package:PerformanceAnalytics':
##
##     textplot
```

```
library(RColorBrewer)
library(devtools)
```

```
## Loading required package: usethis
```

```
library(gtrendsR)
library(plotly)
```

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

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

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

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

```
library(textdata)
library(DT)
library(rvest)
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
library(splusTimeDate)
```

```
##
## Attaching package: 'splusTimeDate'
```

```
## The following objects are masked from 'package:lubridate':
##
##     days, hms, hours, mdy, minutes, seconds, years
```

```
## The following objects are masked from 'package:base':
##
##     months, quarters, sort.list, weekdays
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

Part1:Data Overview

Extract the historical data of selected tickers from yahoo finance package by 'getSymbols' function, modified variable names and index.

```
start = as.Date("2020-01-01")
end = as.Date("2022-03-31")

getSymbols(c("AAPL", "GOOGL", "MSFT", "AMZN", "^GSPC"), src = "yahoo", from = start, to = end)
```

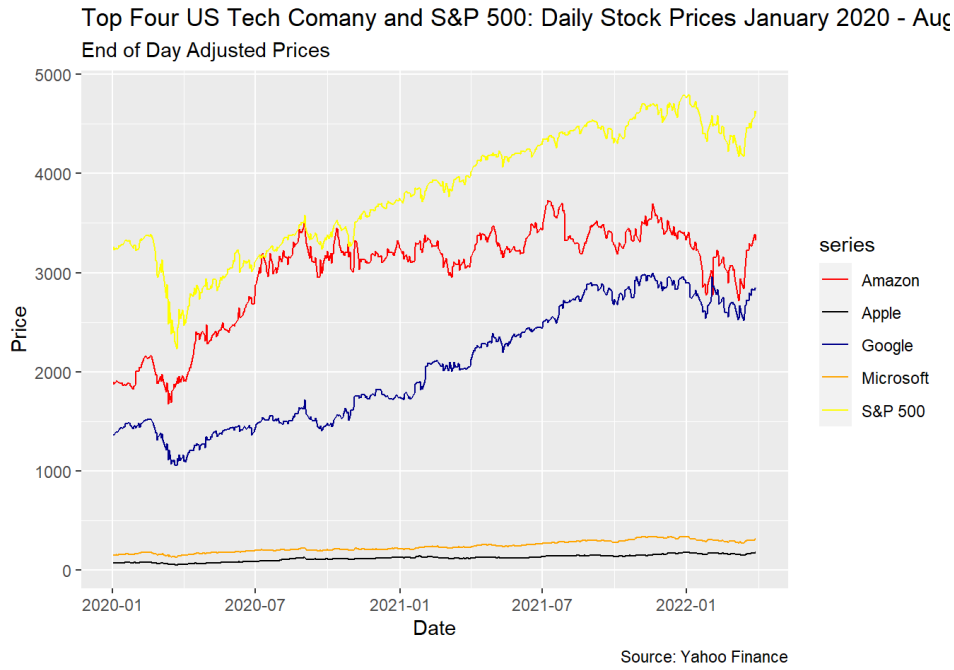
```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "AAPL"  "GOOGL" "MSFT"  "AMZN"  "^GSPC"
```

```
stocks_overview = as.xts(data.frame(A = AAPL[, "AAPL.Adjusted"],
                            B = GOOGL[, "GOOGL.Adjusted"],
                            C = MSFT[, "MSFT.Adjusted"],
                            D = AMZN[, "AMZN.Adjusted"],
                            E = GSPC[,"GSPC.Adjusted"]))
names(stocks_overview) = c("Apple", "Google", "Microsoft", "Amazon", "S&P 500")
index(stocks_overview) = as.Date(index(stocks_overview))
```

Do the visualization of selected stocks

```
stocks_overview_plot = tidy(stocks_overview) %>%

  ggplot(aes(x=index,y=value, color=series)) +
  labs(title = "Top Four US Tech Comany and S&P 500: Daily Stock Prices January 2020 - August 2021",

       subtitle = "End of Day Adjusted Prices",
       caption = " Source: Yahoo Finance") +

  xlab("Date") + ylab("Price") +
  scale_color_manual(values = c("Red", "Black", "DarkBlue","Orange","Yellow"))+
  geom_line()

stocks_overview_plot
```

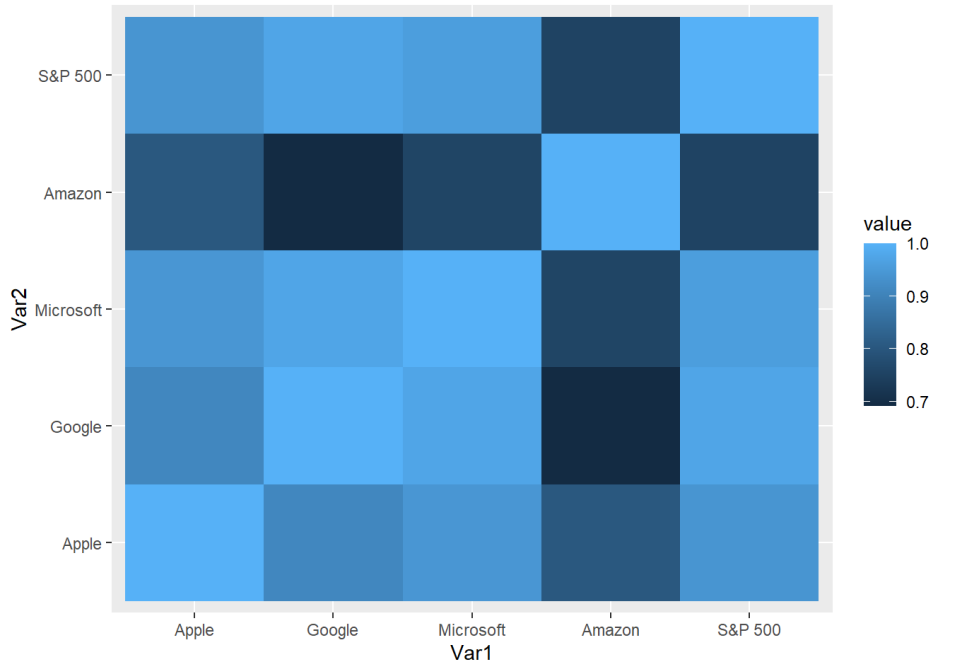


Plot the correlation matrix of selected tickers

```
corr_mat <- as.matrix(stocks_overview) %>% cor()
head(corr_mat)
```

```
##              Apple    Google Microsoft    Amazon   S&P 500
## Apple     1.0000000 0.9106459 0.9431834 0.8026184 0.9397450
## Google    0.9106459 1.0000000 0.9747853 0.6915506 0.9756680
## Microsoft 0.9431834 0.9747853 1.0000000 0.7567049 0.9594247
## Amazon    0.8026184 0.6915506 0.7567049 1.0000000 0.7522188
## S&P 500   0.9397450 0.9756680 0.9594247 0.7522188 1.0000000
```

```
melted_cormat <- melt(corr_mat)

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill = value)) +
  geom_tile()
```



It is quite clear that the correlation between top 4 tech companies is pretty high. If we can make profit of one ticker, we will also have the opportunities to make profit by investing others.

Part2:Sentimental Analysis

Goal: We hope to get insights into the market by conducting sentimental analysis to news headlines. In this case, we are looking for the hotspots of the market and deciding whether our strategies are in right direction.

Preparing the data *We tried a different way to extract the data from the web*

```
t1<- ISOdate(2020,01,01,hour = 0)
as.integer(t1)
```

```
## [1] 1577836800
```

```
t2<- ISOdate(2022,03,31, hour = 0)
as.integer(t2)
```

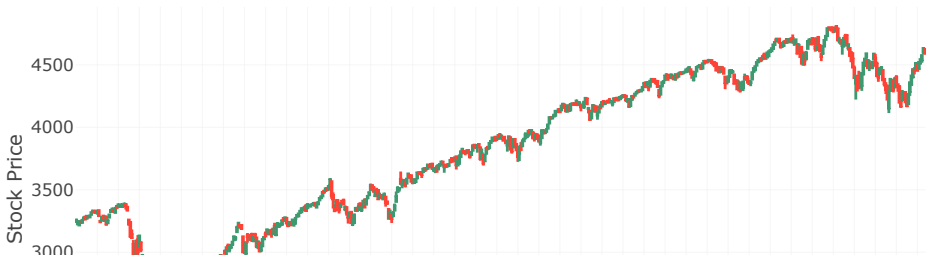
```
## [1] 1648684800
```

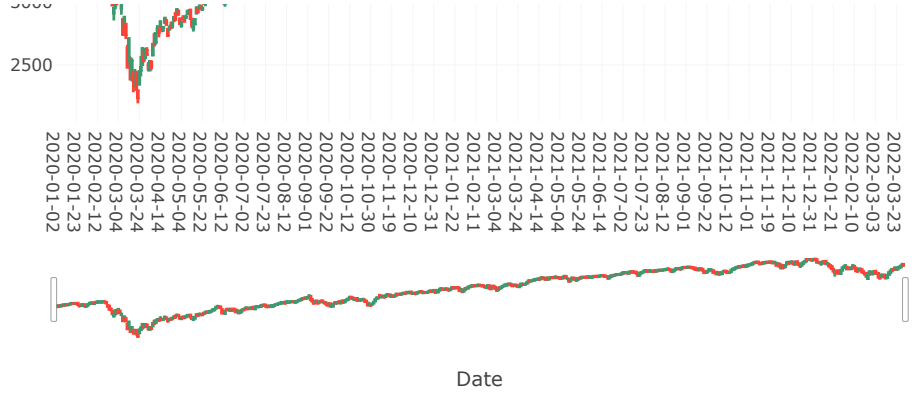
```
stock<- "^GSPC"
url <- paste("https://query1.finance.yahoo.com/v7/finance/download/",
             stock,
             "?period1=",
             as.integer(t1),
             "&period2=",
             as.integer(t2),
             "&interval=1d&events=history",
             sep="")

senti_ticker_data <- read.csv(url)

stock_data <- drop_na(senti_ticker_data)

stock_data$Name <- c("SP500")

headlines_data <- read.csv("india-news-headlines.csv")
```

Visualize the historical data of benchmark we are interested in during the period of pandemic.

```
plot_candlestick <- stock_data %>%
  plot_ly(x = ~Date,
          type = "candlestick",
          open = ~Open,
          close = ~Close,
          high = ~High,
          low = ~Low,
          name = "price") %>%
  layout(rangeslider=list(visible = FALSE), yaxis = list(title = "Stock Price",
                                                         showgrid = TRUE,
                                                         showticklabels = TRUE))

plot_candlestick
```

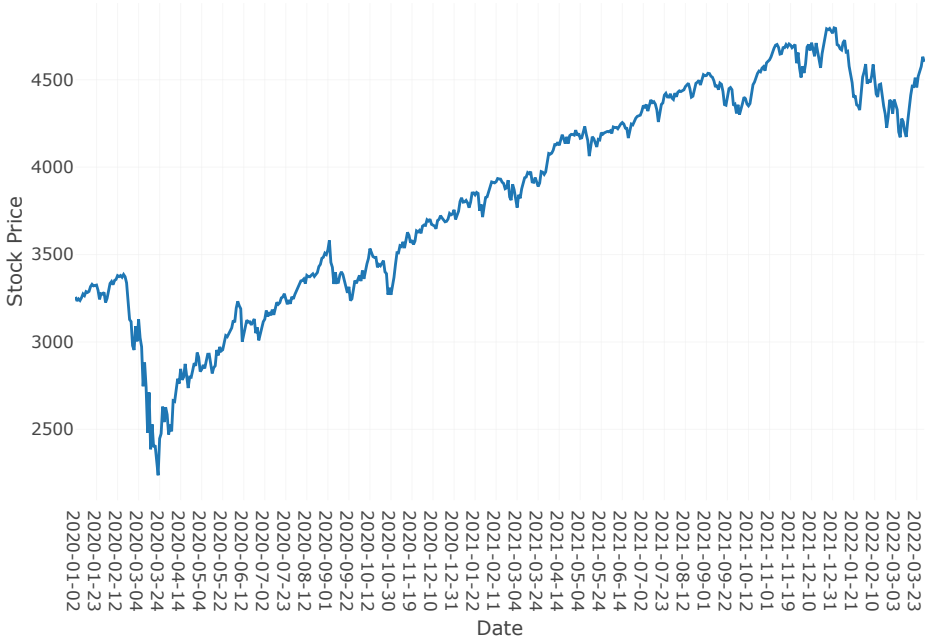
```
## Warning: 'layout' objects don't have these attributes: 'rangeslider'
## Valid attributes include:
## '_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calendar', 'clickmode', 'coloraxis', 'colors
cale', 'colorway', 'computed', 'datarevision', 'dragmode', 'editrevision', 'editType', 'font', 'geo', 'grid', 'height', 'hid
esources', 'hoverdistance', 'hoverlabel', 'hovermode', 'images', 'legend', 'mapbox', 'margin', 'meta', 'metasrc', 'modebar',
'newshape', 'paper_bgcolor', 'plot_bgcolor', 'polar', 'scene', 'selectdirection', 'selectionrevision', 'separators', 'shape
s', 'showlegend', 'sliders', 'spikedistance', 'template', 'ternary', 'title', 'transition', 'uirevision', 'uniformtext', 'up
datemenus', 'width', 'xaxis', 'yaxis', 'barmode', 'bargap', 'mapType'
```





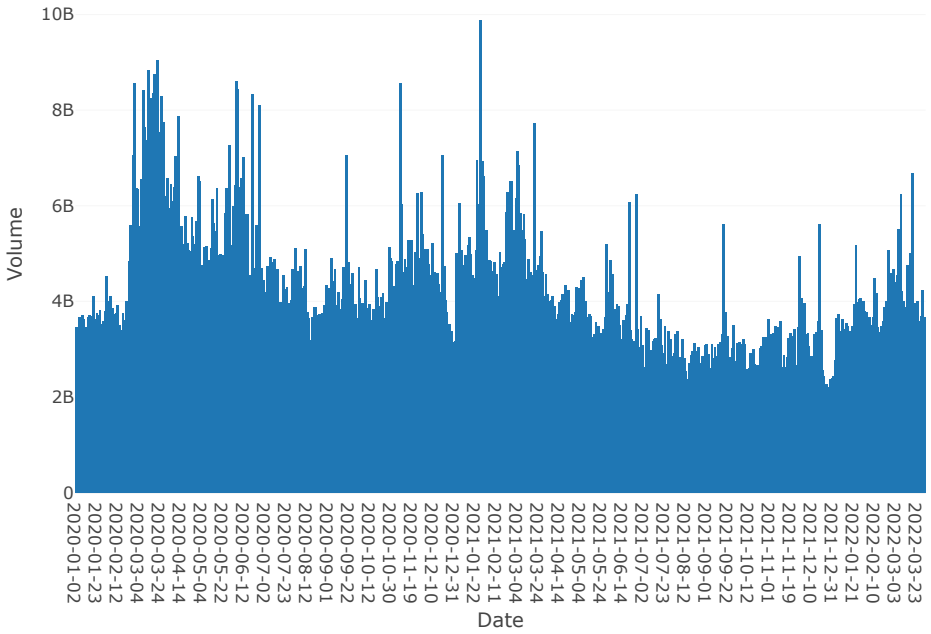

```
plot_scatter <- stock_data %>%
  plot_ly(x = ~Date,y = ~Close, type = "scatter", mode = "lines") %>%
  layout(rangeslider=list(visible = FALSE), yaxis = list(title = "Stock Price",
                                                         showgrid = TRUE,
                                                         showticklabels = TRUE))
plot_scatter
```

```
## Warning: 'layout' objects don't have these attributes: 'rangeslider'
## Valid attributes include:
## '_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calendar', 'clickmode', 'coloraxis', 'colors
cale', 'colorway', 'computed', 'datarevision', 'dragmode', 'editrevision', 'editType', 'font', 'geo', 'grid', 'height', 'hid
esources', 'hoverdistance', 'hoverlabel', 'hovermode', 'images', 'legend', 'mapbox', 'margin', 'meta', 'metasrc', 'modebar',
'newshape', 'paper_bgcolor', 'plot_bgcolor', 'polar', 'scene', 'selectdirection', 'selectionrevision', 'separators', 'shape
s', 'showlegend', 'sliders', 'spikedistance', 'template', 'ternary', 'title', 'transition', 'uirevision', 'uniformtext', 'up
datemenus', 'width', 'xaxis', 'yaxis', 'barmode', 'bargap', 'mapType'
```




```
plot_volume <- stock_data %>%
  plot_ly(x=~Date, y=~Volume, type='bar', Name = "Volume") %>%
  layout(yaxis = list(title = "Volume"))

plot_volume
```

```
## Warning: 'bar' objects don't have these attributes: 'Name'
## Valid attributes include:
## '_deprecated', 'alignmentgroup', 'base', 'basesrc', 'cliponaxis', 'constraintext', 'customdata', 'customdatasrc', 'dx',
'dy', 'error_x', 'error_y', 'hoverinfo', 'hoverinfosrc', 'hoverlabel', 'hovertemplate', 'hovertemplatesrc', 'hovertext', 'ho
vertextsrc', 'ids', 'idssrc', 'insidetextanchor', 'insidetextfont', 'legendgroup', 'legendgrouptitle', 'legendrank', 'marke
r', 'meta', 'metasrc', 'name', 'offset', 'offsetgroup', 'offsetsrc', 'opacity', 'orientation', 'outsidetextfont', 'selecte
d', 'selectedpoints', 'showlegend', 'stream', 'text', 'textangle', 'textfont', 'textposition', 'textpositionsrc', 'textsrc',
'texttemplate', 'texttemplatesrc', 'transforms', 'type', 'uid', 'uirevision', 'unselected', 'visible', 'width', 'widthsrc',
'x', 'x0', 'xaxis', 'xcalendar', 'xhoverformat', 'xperiod', 'xperiod0', 'xperiodalignment', 'xsrc', 'y', 'y0', 'yaxis', 'yca
lendar', 'yhoverformat', 'yperiod', 'yperiod0', 'yperiodalignment', 'ysrc', 'key', 'set', 'frame', 'transforms', '_isNestedK
ey', '_isSimpleKey', '_isGraticule', '_bbox'
```



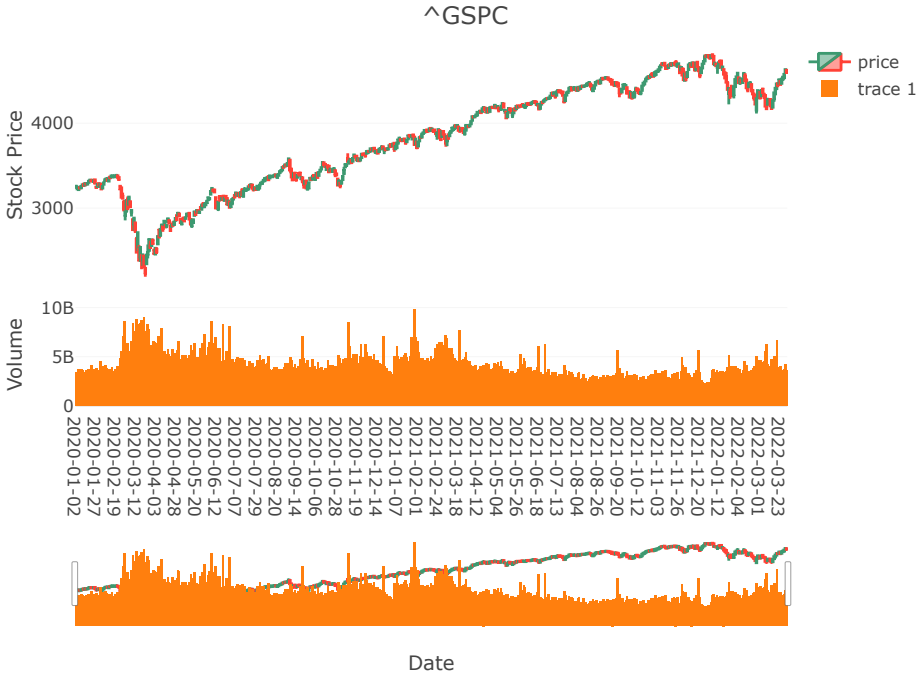

```
plot_combined <- subplot(plot_candlestick, plot_volume, heights = c(0.7,0.3), nrows=2,
                         shareX = TRUE, titleY = TRUE) %>%
  layout(title = paste0(stock))
```

```
## Warning: 'layout' objects don't have these attributes: 'rangeslider'
## Valid attributes include:
## '_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calendar', 'clickmode', 'coloraxis', 'colors
cale', 'colorway', 'computed', 'datarevision', 'dragmode', 'editrevision', 'editType', 'font', 'geo', 'grid', 'height', 'hid
esources', 'hoverdistance', 'hoverlabel', 'hovermode', 'images', 'legend', 'mapbox', 'margin', 'meta', 'metasrc', 'modebar',
'newshape', 'paper_bgcolor', 'plot_bgcolor', 'polar', 'scene', 'selectdirection', 'selectionrevision', 'separators', 'shape
s', 'showlegend', 'sliders', 'spikedistance', 'template', 'ternary', 'title', 'transition', 'uirevision', 'uniformtext', 'up
datemenus', 'width', 'xaxis', 'yaxis', 'barmode', 'bargap', 'mapType'
```

```
## Warning: 'bar' objects don't have these attributes: 'Name'
## Valid attributes include:
## '_deprecated', 'alignmentgroup', 'base', 'basesrc', 'cliponaxis', 'constraintext', 'customdata', 'customdatasrc', 'dx',
'dy', 'error_x', 'error_y', 'hoverinfo', 'hoverinfosrc', 'hoverlabel', 'hovertemplate', 'hovertemplatesrc', 'hovertext', 'ho
vertextsrc', 'ids', 'idssrc', 'insidetextanchor', 'insidetextfont', 'legendgroup', 'legendgrouptitle', 'legendrank', 'marke
r', 'meta', 'metasrc', 'name', 'offset', 'offsetgroup', 'offsetsrc', 'opacity', 'orientation', 'outsidetextfont', 'selecte
d', 'selectedpoints', 'showlegend', 'stream', 'text', 'textangle', 'textfont', 'textposition', 'textpositionsrc', 'textsrc',
'texttemplate', 'texttemplatesrc', 'transforms', 'type', 'uid', 'uirevision', 'unselected', 'visible', 'width', 'widthsrc',
'x', 'x0', 'xaxis', 'xcalendar', 'xhoverformat', 'xperiod', 'xperiod0', 'xperiodalignment', 'xsrc', 'y', 'y0', 'yaxis', 'yca
lendar', 'yhoverformat', 'yperiod', 'yperiod0', 'yperiodalignment', 'ysrc', 'key', 'set', 'frame', 'transforms', '_isNestedK
ey', '_isSimpleKey', '_isGraticule', '_bbox'
```

```
plot_combined
```

```
## Warning: 'layout' objects don't have these attributes: 'rangeslider'
## Valid attributes include:
## '_deprecated', 'activeshape', 'annotations', 'autosize', 'autotypenumbers', 'calendar', 'clickmode', 'coloraxis', 'colors
cale', 'colorway', 'computed', 'datarevision', 'dragmode', 'editrevision', 'editType', 'font', 'geo', 'grid', 'height', 'hid
esources', 'hoverdistance', 'hoverlabel', 'hovermode', 'images', 'legend', 'mapbox', 'margin', 'meta', 'metasrc', 'modebar',
'newshape', 'paper_bgcolor', 'plot_bgcolor', 'polar', 'scene', 'selectdirection', 'selectionrevision', 'separators', 'shape
s', 'showlegend', 'sliders', 'spikedistance', 'template', 'ternary', 'title', 'transition', 'uirevision', 'uniformtext', 'up
datemenus', 'width', 'xaxis', 'yaxis', 'barmode', 'bargap', 'mapType'

## Warning: 'bar' objects don't have these attributes: 'Name'
## Valid attributes include:
## '_deprecated', 'alignmentgroup', 'base', 'basesrc', 'cliponaxis', 'constraintext', 'customdata', 'customdatasrc', 'dx',
'dy', 'error_x', 'error_y', 'hoverinfo', 'hoverinfosrc', 'hoverlabel', 'hovertemplate', 'hovertemplatesrc', 'hovertext', 'ho
vertextsrc', 'ids', 'idssrc', 'insidetextanchor', 'insidetextfont', 'legendgroup', 'legendgrouptitle', 'legendrank', 'marke
r', 'meta', 'metasrc', 'name', 'offset', 'offsetgroup', 'offsetsrc', 'opacity', 'orientation', 'outsidetextfont', 'selecte
d', 'selectedpoints', 'showlegend', 'stream', 'text', 'textangle', 'textfont', 'textposition', 'textpositionsrc', 'textsrc',
'texttemplate', 'texttemplatesrc', 'transforms', 'type', 'uid', 'uirevision', 'unselected', 'visible', 'width', 'widthsrc',
'x', 'x0', 'xaxis', 'xcalendar', 'xhoverformat', 'xperiod', 'xperiod0', 'xperiodalignment', 'xsrc', 'y', 'y0', 'yaxis', 'yca
lendar', 'yhoverformat', 'yperiod', 'yperiod0', 'yperiodalignment', 'ysrc', 'key', 'set', 'frame', 'transforms', '_isNestedK
ey', '_isSimpleKey', '_isGraticule', '_bbox'
```



In the beginning of the 2020, we observed a huge plunge in stock price. However, the market still has a great number of volume trading the benchmark. Such liquidity of the market excites us to do further analysis.

We first use the gtrends package to visualize the interest over time(hits by google search) for the selected stock over the years.

It is now possible to view the relationship between interest over time ('hits') and stock performance. A left join is used to combine trend and stock data by date. The outcome of the join is used to visualize the relationship between hits and stock close prices.

```
data("countries")

keyword_hits <- gtrends(keyword = stock, geo = "US", onlyInterest = TRUE)
head(keyword_hits)
```

```
## $interest_over_time
##           date hits keyword geo      time gprop category
## 1   2017-04-30    0   ^GSPC  US today+5-y   web        0
## 2   2017-05-07    3   ^GSPC  US today+5-y   web        0
## 3   2017-05-14    7   ^GSPC  US today+5-y   web        0
## 4   2017-05-21    3   ^GSPC  US today+5-y   web        0
## 5   2017-05-28    4   ^GSPC  US today+5-y   web        0
## 6   2017-06-04    3   ^GSPC  US today+5-y   web        0
## 7   2017-06-11    0   ^GSPC  US today+5-y   web        0
## 8   2017-06-18    3   ^GSPC  US today+5-y   web        0
## 9   2017-06-25    0   ^GSPC  US today+5-y   web        0
## 10  2017-07-02    0   ^GSPC  US today+5-y   web        0
## 11  2017-07-09    0   ^GSPC  US today+5-y   web        0
## 12  2017-07-16    0   ^GSPC  US today+5-y   web        0
## 13  2017-07-23    3   ^GSPC  US today+5-y   web        0
## 14  2017-07-30    0   ^GSPC  US today+5-y   web        0
## 15  2017-08-06    0   ^GSPC  US today+5-y   web        0
## 16  2017-08-13    0   ^GSPC  US today+5-y   web        0
## 17  2017-08-20    0   ^GSPC  US today+5-y   web        0
## 18  2017-08-27    0   ^GSPC  US today+5-y   web        0
## 19  2017-09-03    0   ^GSPC  US today+5-y   web        0
## 20  2017-09-10    3   ^GSPC  US today+5-y   web        0
## 21  2017-09-17    0   ^GSPC  US today+5-y   web        0
## 22  2017-09-24    6   ^GSPC  US today+5-y   web        0
## 23  2017-10-01    3   ^GSPC  US today+5-y   web        0
## 24  2017-10-08    3   ^GSPC  US today+5-y   web        0
## 25  2017-10-15    0   ^GSPC  US today+5-y   web        0
## 26  2017-10-22    0   ^GSPC  US today+5-y   web        0
## 27  2017-10-29    6   ^GSPC  US today+5-y   web        0
## 28  2017-11-05    6   ^GSPC  US today+5-y   web        0
## 29  2017-11-12    0   ^GSPC  US today+5-y   web        0
## 30  2017-11-19    0   ^GSPC  US today+5-y   web        0
## 31  2017-11-26    3   ^GSPC  US today+5-y   web        0
## 32  2017-12-03    3   ^GSPC  US today+5-y   web        0
## 33  2017-12-10    3   ^GSPC  US today+5-y   web        0
## 34  2017-12-17    0   ^GSPC  US today+5-y   web        0
## 35  2017-12-24    4   ^GSPC  US today+5-y   web        0
## 36  2017-12-31    0   ^GSPC  US today+5-y   web        0
## 37  2018-01-07    9   ^GSPC  US today+5-y   web        0
## 38  2018-01-14    3   ^GSPC  US today+5-y   web        0
## 39  2018-01-21    6   ^GSPC  US today+5-y   web        0
## 40  2018-01-28    3   ^GSPC  US today+5-y   web        0
## 41  2018-02-04    6   ^GSPC  US today+5-y   web        0
## 42  2018-02-11    6   ^GSPC  US today+5-y   web        0
## 43  2018-02-18    6   ^GSPC  US today+5-y   web        0
## 44  2018-02-25    3   ^GSPC  US today+5-y   web        0
## 45  2018-03-04    6   ^GSPC  US today+5-y   web        0
## 46  2018-03-11   36   ^GSPC  US today+5-y   web        0
## 47  2018-03-18   33   ^GSPC  US today+5-y   web        0
## 48  2018-03-25   53   ^GSPC  US today+5-y   web        0
## 49  2018-04-01   34   ^GSPC  US today+5-y   web        0
## 50  2018-04-08   64   ^GSPC  US today+5-y   web        0
## 51  2018-04-15   46   ^GSPC  US today+5-y   web        0
## 52  2018-04-22   49   ^GSPC  US today+5-y   web        0
## 53  2018-04-29   58   ^GSPC  US today+5-y   web        0
## 54  2018-05-06   43   ^GSPC  US today+5-y   web        0
## 55  2018-05-13   47   ^GSPC  US today+5-y   web        0
## 56  2018-05-20   25   ^GSPC  US today+5-y   web        0
## 57  2018-05-27   58   ^GSPC  US today+5-y   web        0
## 58  2018-06-03   48   ^GSPC  US today+5-y   web        0
## 59  2018-06-10   25   ^GSPC  US today+5-y   web        0
## 60  2018-06-17   48   ^GSPC  US today+5-y   web        0
## 61  2018-06-24   32   ^GSPC  US today+5-y   web        0
## 62  2018-07-01   43   ^GSPC  US today+5-y   web        0
## 63  2018-07-08   32   ^GSPC  US today+5-y   web        0
## 64  2018-07-15   35   ^GSPC  US today+5-y   web        0
## 65  2018-07-22   26   ^GSPC  US today+5-y   web        0
## 66  2018-07-29   32   ^GSPC  US today+5-y   web        0
## 67  2018-08-05   71   ^GSPC  US today+5-y   web        0
## 68  2018-08-12  100   ^GSPC  US today+5-y   web        0
## 69  2018-08-19   71   ^GSPC  US today+5-y   web        0
## 70  2018-08-26   95   ^GSPC  US today+5-y   web        0
## 71  2018-09-02   92   ^GSPC  US today+5-y   web        0
## 72  2018-09-09   46   ^GSPC  US today+5-y   web        0
```

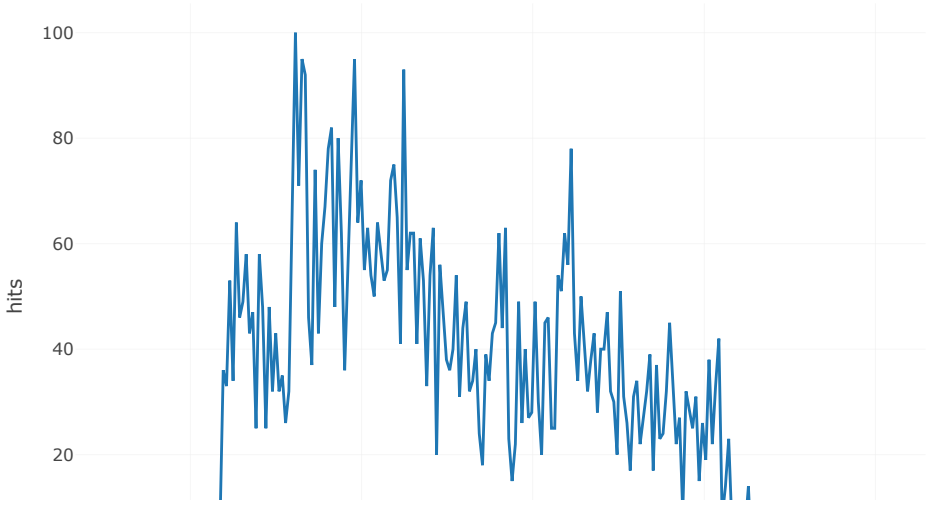
```
## 73  2018-09-16   37   ^GSPC  US today+5-y   web        0
## 74  2018-09-23   74   ^GSPC  US today+5-y   web        0
## 75  2018-09-30   43   ^GSPC  US today+5-y   web        0
## 76  2018-10-07   60   ^GSPC  US today+5-y   web        0
## 77  2018-10-14   67   ^GSPC  US today+5-y   web        0
## 78  2018-10-21   78   ^GSPC  US today+5-y   web        0
## 79  2018-10-28   82   ^GSPC  US today+5-y   web        0
## 80  2018-11-04   48   ^GSPC  US today+5-y   web        0
## 81  2018-11-11   80   ^GSPC  US today+5-y   web        0
## 82  2018-11-18   62   ^GSPC  US today+5-y   web        0
## 83  2018-11-25   36   ^GSPC  US today+5-y   web        0
## 84  2018-12-02   55   ^GSPC  US today+5-y   web        0
## 85  2018-12-09   74   ^GSPC  US today+5-y   web        0
## 86  2018-12-16   95   ^GSPC  US today+5-y   web        0
## 87  2018-12-23   64   ^GSPC  US today+5-y   web        0
## 88  2018-12-30   72   ^GSPC  US today+5-y   web        0
## 89  2019-01-06   55   ^GSPC  US today+5-y   web        0
## 90  2019-01-13   63   ^GSPC  US today+5-y   web        0
## 91  2019-01-20   54   ^GSPC  US today+5-y   web        0
## 92  2019-01-27   50   ^GSPC  US today+5-y   web        0
## 93  2019-02-03   64   ^GSPC  US today+5-y   web        0
## 94  2019-02-10   58   ^GSPC  US today+5-y   web        0
## 95  2019-02-17   53   ^GSPC  US today+5-y   web        0
## 96  2019-02-24   55   ^GSPC  US today+5-y   web        0
## 97  2019-03-03   72   ^GSPC  US today+5-y   web        0
## 98  2019-03-10   75   ^GSPC  US today+5-y   web        0
## 99  2019-03-17   65   ^GSPC  US today+5-y   web        0
## 100 2019-03-24   41   ^GSPC  US today+5-y   web        0
## 101 2019-03-31   93   ^GSPC  US today+5-y   web        0
## 102 2019-04-07   55   ^GSPC  US today+5-y   web        0
## 103 2019-04-14   62   ^GSPC  US today+5-y   web        0
## 104 2019-04-21   62   ^GSPC  US today+5-y   web        0
## 105 2019-04-28   41   ^GSPC  US today+5-y   web        0
## 106 2019-05-05   61   ^GSPC  US today+5-y   web        0
## 107 2019-05-12   53   ^GSPC  US today+5-y   web        0
## 108 2019-05-19   33   ^GSPC  US today+5-y   web        0
## 109 2019-05-26   54   ^GSPC  US today+5-y   web        0
## 110 2019-06-02   63   ^GSPC  US today+5-y   web        0
## 111 2019-06-09   20   ^GSPC  US today+5-y   web        0
## 112 2019-06-16   56   ^GSPC  US today+5-y   web        0
## 113 2019-06-23   47   ^GSPC  US today+5-y   web        0
## 114 2019-06-30   38   ^GSPC  US today+5-y   web        0
## 115 2019-07-07   36   ^GSPC  US today+5-y   web        0
## 116 2019-07-14   40   ^GSPC  US today+5-y   web        0
## 117 2019-07-21   54   ^GSPC  US today+5-y   web        0
## 118 2019-07-28   31   ^GSPC  US today+5-y   web        0
## 119 2019-08-04   44   ^GSPC  US today+5-y   web        0
## 120 2019-08-11   49   ^GSPC  US today+5-y   web        0
## 121 2019-08-18   32   ^GSPC  US today+5-y   web        0
## 122 2019-08-25   34   ^GSPC  US today+5-y   web        0
## 123 2019-09-01   40   ^GSPC  US today+5-y   web        0
## 124 2019-09-08   24   ^GSPC  US today+5-y   web        0
## 125 2019-09-15   18   ^GSPC  US today+5-y   web        0
## 126 2019-09-22   39   ^GSPC  US today+5-y   web        0
## 127 2019-09-29   34   ^GSPC  US today+5-y   web        0
## 128 2019-10-06   43   ^GSPC  US today+5-y   web        0
## 129 2019-10-13   45   ^GSPC  US today+5-y   web        0
## 130 2019-10-20   62   ^GSPC  US today+5-y   web        0
## 131 2019-10-27   44   ^GSPC  US today+5-y   web        0
## 132 2019-11-03   63   ^GSPC  US today+5-y   web        0
## 133 2019-11-10   23   ^GSPC  US today+5-y   web        0
## 134 2019-11-17   15   ^GSPC  US today+5-y   web        0
## 135 2019-11-24   22   ^GSPC  US today+5-y   web        0
## 136 2019-12-01   49   ^GSPC  US today+5-y   web        0
## 137 2019-12-08   26   ^GSPC  US today+5-y   web        0
## 138 2019-12-15   40   ^GSPC  US today+5-y   web        0
## 139 2019-12-22   27   ^GSPC  US today+5-y   web        0
## 140 2019-12-29   28   ^GSPC  US today+5-y   web        0
## 141 2020-01-05   49   ^GSPC  US today+5-y   web        0
## 142 2020-01-12   30   ^GSPC  US today+5-y   web        0
## 143 2020-01-19   20   ^GSPC  US today+5-y   web        0
## 144 2020-01-26   45   ^GSPC  US today+5-y   web        0
## 145 2020-02-02   46   ^GSPC  US today+5-y   web        0
## 146 2020-02-09   25   ^GSPC  US today+5-y   web        0
```

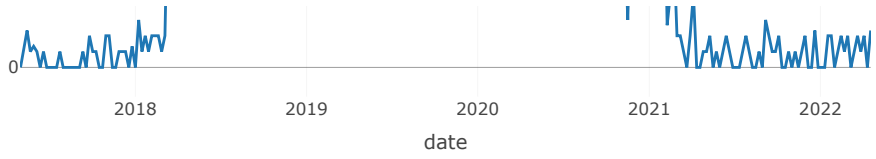
```
## 147 2020-02-16   25   ^GSPC  US today+5-y   web        0
## 148 2020-02-23   54   ^GSPC  US today+5-y   web        0
## 149 2020-03-01   51   ^GSPC  US today+5-y   web        0
## 150 2020-03-08   62   ^GSPC  US today+5-y   web        0
## 151 2020-03-15   56   ^GSPC  US today+5-y   web        0
## 152 2020-03-22   78   ^GSPC  US today+5-y   web        0
## 153 2020-03-29   43   ^GSPC  US today+5-y   web        0
## 154 2020-04-05   34   ^GSPC  US today+5-y   web        0
## 155 2020-04-12   50   ^GSPC  US today+5-y   web        0
## 156 2020-04-19   40   ^GSPC  US today+5-y   web        0
## 157 2020-04-26   32   ^GSPC  US today+5-y   web        0
## 158 2020-05-03   38   ^GSPC  US today+5-y   web        0
## 159 2020-05-10   43   ^GSPC  US today+5-y   web        0
## 160 2020-05-17   28   ^GSPC  US today+5-y   web        0
## 161 2020-05-24   40   ^GSPC  US today+5-y   web        0
## 162 2020-05-31   40   ^GSPC  US today+5-y   web        0
## 163 2020-06-07   47   ^GSPC  US today+5-y   web        0
## 164 2020-06-14   32   ^GSPC  US today+5-y   web        0
## 165 2020-06-21   30   ^GSPC  US today+5-y   web        0
## 166 2020-06-28   20   ^GSPC  US today+5-y   web        0
## 167 2020-07-05   51   ^GSPC  US today+5-y   web        0
## 168 2020-07-12   31   ^GSPC  US today+5-y   web        0
## 169 2020-07-19   26   ^GSPC  US today+5-y   web        0
## 170 2020-07-26   17   ^GSPC  US today+5-y   web        0
## 171 2020-08-02   31   ^GSPC  US today+5-y   web        0
## 172 2020-08-09   34   ^GSPC  US today+5-y   web        0
## 173 2020-08-16   22   ^GSPC  US today+5-y   web        0
## 174 2020-08-23   27   ^GSPC  US today+5-y   web        0
## 175 2020-08-30   32   ^GSPC  US today+5-y   web        0
## 176 2020-09-06   39   ^GSPC  US today+5-y   web        0
## 177 2020-09-13   17   ^GSPC  US today+5-y   web        0
## 178 2020-09-20   37   ^GSPC  US today+5-y   web        0
## 179 2020-09-27   23   ^GSPC  US today+5-y   web        0
## 180 2020-10-04   24   ^GSPC  US today+5-y   web        0
## 181 2020-10-11   32   ^GSPC  US today+5-y   web        0
## 182 2020-10-18   45   ^GSPC  US today+5-y   web        0
## 183 2020-10-25   35   ^GSPC  US today+5-y   web        0
## 184 2020-11-01   22   ^GSPC  US today+5-y   web        0
## 185 2020-11-08   27   ^GSPC  US today+5-y   web        0
## 186 2020-11-15    9   ^GSPC  US today+5-y   web        0
## 187 2020-11-22   32   ^GSPC  US today+5-y   web        0
## 188 2020-11-29   28   ^GSPC  US today+5-y   web        0
## 189 2020-12-06   25   ^GSPC  US today+5-y   web        0
## 190 2020-12-13   31   ^GSPC  US today+5-y   web        0
## 191 2020-12-20   15   ^GSPC  US today+5-y   web        0
## 192 2020-12-27   26   ^GSPC  US today+5-y   web        0
## 193 2021-01-03   19   ^GSPC  US today+5-y   web        0
## 194 2021-01-10   38   ^GSPC  US today+5-y   web        0
## 195 2021-01-17   22   ^GSPC  US today+5-y   web        0
## 196 2021-01-24   33   ^GSPC  US today+5-y   web        0
## 197 2021-01-31   42   ^GSPC  US today+5-y   web        0
## 198 2021-02-07    8   ^GSPC  US today+5-y   web        0
## 199 2021-02-14   14   ^GSPC  US today+5-y   web        0
## 200 2021-02-21   23   ^GSPC  US today+5-y   web        0
## 201 2021-02-28    6   ^GSPC  US today+5-y   web        0
## 202 2021-03-07    6   ^GSPC  US today+5-y   web        0
## 203 2021-03-14    3   ^GSPC  US today+5-y   web        0
## 204 2021-03-21    0   ^GSPC  US today+5-y   web        0
## 205 2021-03-28    6   ^GSPC  US today+5-y   web        0
## 206 2021-04-04   14   ^GSPC  US today+5-y   web        0
## 207 2021-04-11    0   ^GSPC  US today+5-y   web        0
## 208 2021-04-18    0   ^GSPC  US today+5-y   web        0
## 209 2021-04-25    3   ^GSPC  US today+5-y   web        0
## 210 2021-05-02    3   ^GSPC  US today+5-y   web        0
## 211 2021-05-09    6   ^GSPC  US today+5-y   web        0
## 212 2021-05-16    0   ^GSPC  US today+5-y   web        0
## 213 2021-05-23    3   ^GSPC  US today+5-y   web        0
## 214 2021-05-30    0   ^GSPC  US today+5-y   web        0
## 215 2021-06-06    3   ^GSPC  US today+5-y   web        0
## 216 2021-06-13    6   ^GSPC  US today+5-y   web        0
## 217 2021-06-20    3   ^GSPC  US today+5-y   web        0
## 218 2021-06-27    0   ^GSPC  US today+5-y   web        0
## 219 2021-07-04    0   ^GSPC  US today+5-y   web        0
## 220 2021-07-11    0   ^GSPC  US today+5-y   web        0
```

```
## 221 2021-07-18    3   ^GSPC  US today+5-y   web        0
## 222 2021-07-25    6   ^GSPC  US today+5-y   web        0
## 223 2021-08-01    3   ^GSPC  US today+5-y   web        0
## 224 2021-08-08    0   ^GSPC  US today+5-y   web        0
## 225 2021-08-15    0   ^GSPC  US today+5-y   web        0
## 226 2021-08-22    3   ^GSPC  US today+5-y   web        0
## 227 2021-08-29    0   ^GSPC  US today+5-y   web        0
## 228 2021-09-05    9   ^GSPC  US today+5-y   web        0
## 229 2021-09-12    6   ^GSPC  US today+5-y   web        0
## 230 2021-09-19    3   ^GSPC  US today+5-y   web        0
## 231 2021-09-26    3   ^GSPC  US today+5-y   web        0
## 232 2021-10-03    6   ^GSPC  US today+5-y   web        0
## 233 2021-10-10    0   ^GSPC  US today+5-y   web        0
## 234 2021-10-17    0   ^GSPC  US today+5-y   web        0
## 235 2021-10-24    3   ^GSPC  US today+5-y   web        0
## 236 2021-10-31    0   ^GSPC  US today+5-y   web        0
## 237 2021-11-07    3   ^GSPC  US today+5-y   web        0
## 238 2021-11-14    0   ^GSPC  US today+5-y   web        0
## 239 2021-11-21    3   ^GSPC  US today+5-y   web        0
## 240 2021-11-28    6   ^GSPC  US today+5-y   web        0
## 241 2021-12-05    0   ^GSPC  US today+5-y   web        0
## 242 2021-12-12    0   ^GSPC  US today+5-y   web        0
## 243 2021-12-19    7   ^GSPC  US today+5-y   web        0
## 244 2021-12-26    0   ^GSPC  US today+5-y   web        0
## 245 2022-01-02    0   ^GSPC  US today+5-y   web        0
## 246 2022-01-09    0   ^GSPC  US today+5-y   web        0
## 247 2022-01-16    6   ^GSPC  US today+5-y   web        0
## 248 2022-01-23    6   ^GSPC  US today+5-y   web        0
## 249 2022-01-30    0   ^GSPC  US today+5-y   web        0
## 250 2022-02-06    3   ^GSPC  US today+5-y   web        0
## 251 2022-02-13    6   ^GSPC  US today+5-y   web        0
## 252 2022-02-20    3   ^GSPC  US today+5-y   web        0
## 253 2022-02-27    6   ^GSPC  US today+5-y   web        0
## 254 2022-03-06    0   ^GSPC  US today+5-y   web        0
## 255 2022-03-13    3   ^GSPC  US today+5-y   web        0
## 256 2022-03-20    6   ^GSPC  US today+5-y   web        0
## 257 2022-03-27    3   ^GSPC  US today+5-y   web        0
## 258 2022-04-03    6   ^GSPC  US today+5-y   web        0
## 259 2022-04-10    0   ^GSPC  US today+5-y   web        0
## 260 2022-04-17    7   ^GSPC  US today+5-y   web        0
```

```
keyword_hits <- keyword_hits$interest_over_time %>%
  as_tibble() %>%
  select(c(date, hits, keyword))
keyword_hits$date <- as_date(ceiling_date(keyword_hits$date, unit = "weeks", change_on_boundary = NULL, week_start = getOpti
on("lubridate.week.start",1)))
keyword_hits %>%
  plot_ly(x=~date, y=~hits, mode= 'lines', name = "Google Search keyword_hits")
```

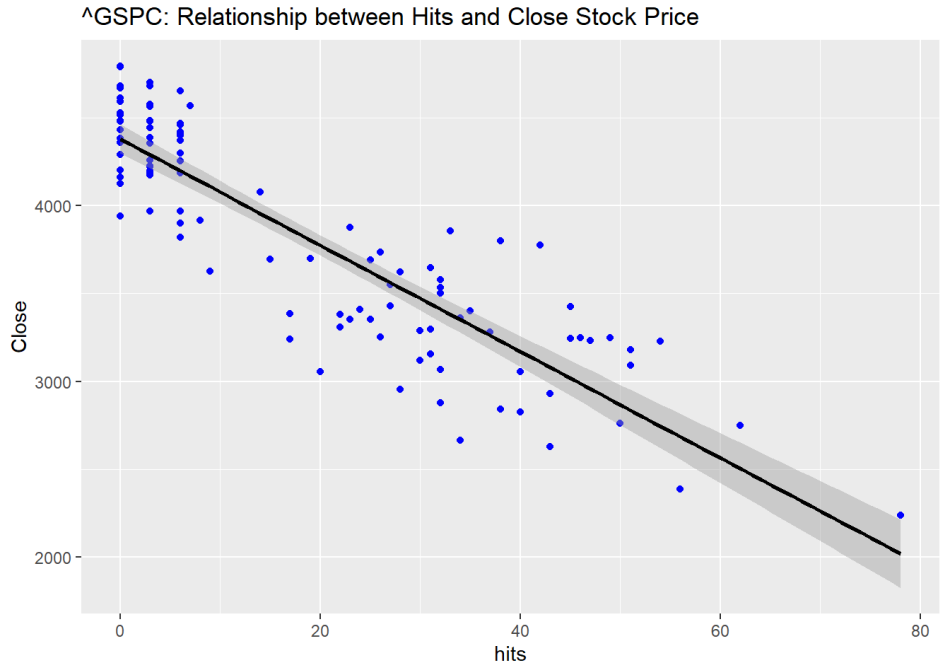
```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter
```






```
#Relation between hits and stock price

keyword_hits<- rename(keyword_hits, Date= date)
stock_data$Date <- as_date(stock_data$Date)
keyword_hits %>%
  left_join(stock_data, by= "Date") %>%
  select(one_of(c("Date", "hits", "Close"))) %>%
  drop_na() %>%
  ggplot(aes(hits, Close))+
  geom_point(color= "blue")+
  geom_smooth(method = lm, color= "black") +
  labs(title =paste0(stock,": Relationship between Hits and Close Stock Price"))
```

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



Discussion: As we can observe from the graph above, the lower the price, the more hits of google search. This means the investments in pandemic period are highly valued by the market.

Write tidy data into csv and save it to the directory

```
write.table(keyword_hits,file = "keyword_hits.csv",sep = ",",col.names = TRUE)
```

After performing the analysis of the google search, we are interested in how the market is reflected from the news headlines. News articles give us excellent insights into the stock market. In the next step, we conduct sentimental analysis on news headlines during the pandemic.

Firstly, we unnest each word in the news articles, and get a bulk of words. We created a word cloud to do a quick visualization of the most frequently used words in the news headlines. afinn sentiment lexicon

We use the afinn sentiment lexicon to assign a score to each word on a scale of -5 to 5. For analysis convenience, we grouped the data by articles and dates to summarise the score by taking the mean for each group.

```
news_article <- headlines_data
news_article<-news_article %>%
  filter(str_detect(headline_text,'STOCK|Stock|stock|STOCKS|Stocks|stocks'))
news_article <- transform(news_article, publish_date = as.Date(as.character(publish_date), "%Y%m%d"))
news_article<- news_article[-c(4)]
news_article<-news_article %>%
  filter(publish_date>'2020-01-01')
news_article<-news_article %>%
  filter(publish_date<'2022-03-31')
news_words <- news_article %>%
  select(c("publish_date","headline_category","headline_text"))%>%
  unnest_tokens(word, headline_text) %>%
  filter(!word %in% append(stop_words$word, values = "chars"), str_detect(word, "^[a-z']+$"))
news_words$publish_date = as_date(news_words$publish_date)
words_only<- dplyr::count(news_words,word)
set.seed(1)
wordcloud(words = words_only$word, freq = words_only$n, scale = c(5,.5), max.words = 50, colors = brewer.pal(8, "Dark2"))
```


```
afinn<- get_sentiments("afinn")

sentiment_analysis <- news_words %>%
  left_join(afinn) %>%
  filter(!is.na(value)) %>%
  group_by(headline_category, publish_date) %>%
  summarise(value= mean(value)) %>%
  mutate(sentiment= ifelse(value>0, "positive", "negative"))
```

```
## Joining, by = "word"
## `summarise()` has grouped output by 'headline_category'. You can override using
## the `.groups` argument.
```

```
datatable(sentiment_analysis)
```

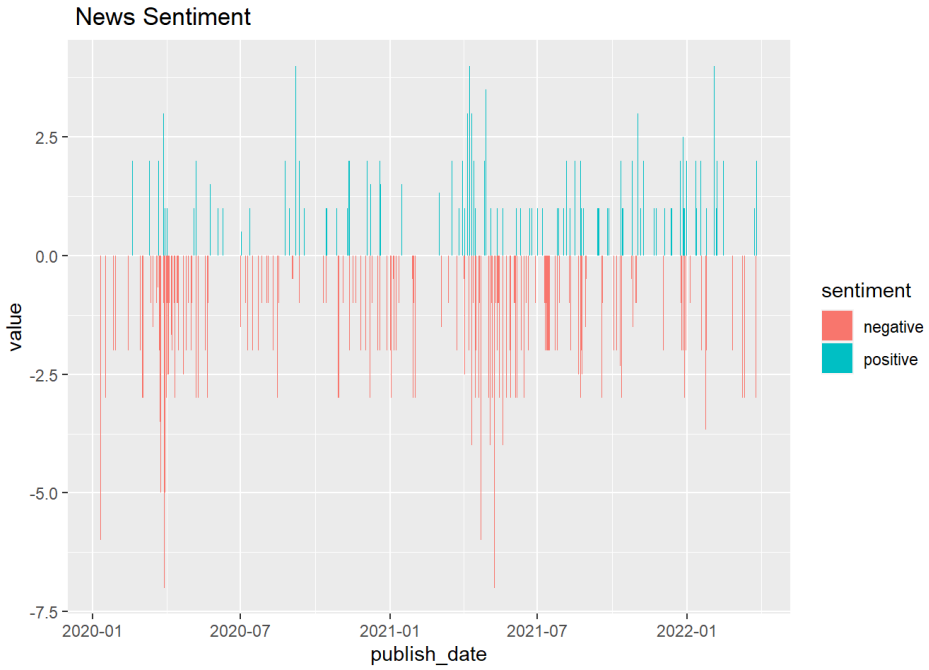
Show 10 entries Search:

	headline_category	publish_date	value	sentiment
1	business.india-business	2020-03-21	-0.666666666666667	negative
2	business.india-business	2020-03-24	-0.5	negative
3	business.india-business	2020-03-25	-2	negative
4	business.india-business	2020-03-28	-1	negative

	headline_category	publish_date	value	sentiment
5	business.india-business	2020-04-08	-2	negative
6	business.india-business	2020-04-22	-2	negative
7	business.india-business	2020-04-25	-2	negative
8	business.india-business	2020-08-04	-1	negative
9	business.india-business	2020-09-11	2	positive
10	business.india-business	2020-10-30	-1	negative

Showing 1 to 10 of 290 entries Previous 1 2 3 4 5 … 29 Next

```
ggplot(sentiment_analysis, aes(publish_date, value)) + geom_bar(stat = "identity", aes(fill=sentiment))  + ggtitle(paste0("
 News Sentiment"))
```



Discussion: We can observed that most words in the headlines are the negative, and the most frequent words along with stocks are mostly about the pandemic. It is reasonable to look for the investment opportunities during this periods.

Part3:Portfolio Contruction

After the analysis above, we decided to invest in the market. Given the market condition, which is pretty negativeWe, we are now constructing portfolios by the skewness and kurtosis as the factors in the model. We firstly use the historical data to find the historical skewness and kurtosis on dataset, and then we use the skewness calculated to regress the kurtosis variables. After having these two coefficients vectors, we can continue backtesting the model and calculating several important criteria in portfolio theory. Finally, we have to test the statistical significance of each factor.

We ranked top 30 stocks in the components of S&P500 benchmark by their market capitalization, and stored them Stockin SP500.csv file. Import the file and tickers, use getSymbols function to get the stock information

```
stock_pool = read.csv("SP500.csv",header = TRUE,fill = FALSE)
stock_port = subset(stock_pool,select = Ticker)
stock_weights = subset(stock_pool,select = Sharesout)
symbols = as.vector(stock_port$Ticker)

ticker_data <- xts()


n <- length(symbols)
pbar <- txtProgressBar(min = 0, max = n, style = 3)
```

```
##
  |
  |                                                                      |   0%
```

```
for(i in 1:length(symbols)){
  symbols[i] -> symbol
  tryit <- try(getSymbols(symbol, from = "2010-04-01", to = "2021-08-31", src = 'yahoo'))
  if(inherits(tryit, "try-error")){
    i <- i+1
  }
  else{
    data <- getSymbols(symbol, from= "2010-04-01",to = "2021-08-31", src = 'yahoo')
    ticker_data <- merge(ticker_data, Ad(get(symbols[i])))
    rm(symbol)
  }
  setTxtProgressBar(pbar,i)
}
```

```
##
  |
  |==                                                                |   3%
  |
  |=====                                                             |   7%
  |
  |=======                                                           |  10%
  |
  |=========                                                         |  13%
  |
  |============                                                      |  17%
  |
  |==============                                                    |  20%
  |
  |================                                                  |  23%
  |
  |===================                                               |  27%
  |
  |=====================                                             |  30%
  |
  |=======================                                           |  33%
  |
  |==========================                                        |  37%
  |
  |============================                                      |  40%
  |
  |==============================                                    |  43%
  |
  |================================                                  |  47%
  |
  |===================================                               |  50%
  |
  |=====================================                             |  53%
  |
  |========================================                          |  57%
  |
  |==========================================                        |  60%
  |
  |============================================                      |  63%
  |
  |==============================================                    |  67%
  |
  |================================================                  |  70%
  |
  |==================================================                |  73%
  |
  |====================================================              |  77%
  |
  |======================================================            |  80%
  |
  |========================================================          |  83%
  |
  |==========================================================        |  87%
  |
  |============================================================      |  90%
  |
  |==============================================================    |  93%
  |
  |================================================================  |  97%
  |
  |==================================================================| 100%
```

```
summary(ticker_data)
```

```
##      Index                    MSFT.Adjusted    AAPL.Adjusted
##  Min.   :2010-03-31 20:00:00   Min.   : 17.90   Min.   :  7.213
##  1st Qu.:2013-02-07 01:00:00   1st Qu.: 25.49   1st Qu.: 16.437
##  Median :2015-12-14 07:00:00   Median : 46.41   Median : 25.942
##  Mean   :2015-12-15 03:31:03   Mean   : 75.84   Mean   : 38.248
##  3rd Qu.:2018-10-21 02:00:00   3rd Qu.:103.13   3rd Qu.: 45.287
##  Max.   :2021-08-29 20:00:00   Max.   :303.47   Max.   :152.702
##  AMZN.Adjusted    GOOG.Adjusted    BRK.B.Adjusted    JPM.Adjusted
##  Min.   : 108.6   Min.   : 217.2   Min.   : 66.00   Min.   : 21.25
##  1st Qu.: 260.3   1st Qu.: 385.6   1st Qu.: 97.73   1st Qu.: 36.92
##  Median : 575.1   Median : 700.7   Median :143.51   Median : 53.71
##  Mean   :1016.0   Mean   : 822.2   Mean   :150.10   Mean   : 66.56
##  3rd Qu.:1695.3   3rd Qu.:1116.3   3rd Qu.:198.48   3rd Qu.: 95.11
##  Max.   :3731.4   Max.   :2909.4   Max.   :292.52   Max.   :162.31
##   JNJ.Adjusted     WMT.Adjusted     XOM.Adjusted     V.Adjusted
##  Min.   : 40.48   Min.   : 36.53   Min.   :27.53   Min.   : 15.09
##  1st Qu.: 58.56   1st Qu.: 56.87   1st Qu.:53.82   1st Qu.: 37.16
##  Median : 87.77   Median : 64.43   Median :60.30   Median : 71.35
##  Mean   : 93.56   Mean   : 75.09   Mean   :57.83   Mean   : 90.04
##  3rd Qu.:122.71   3rd Qu.: 91.16   3rd Qu.:64.51   3rd Qu.:136.71
##  Max.   :176.12   Max.   :150.26   Max.   :73.20   Max.   :249.74
##   PG.Adjusted      BAC.Adjusted     MA.Adjusted       T.Adjusted
##  Min.   : 41.39   Min.   : 4.297   Min.   : 18.06   Min.   : 9.142
##  1st Qu.: 56.97   1st Qu.:11.755   1st Qu.: 49.16   1st Qu.:15.502
##  Median : 68.24   Median :14.688   Median : 90.56   Median :17.557
##  Mean   : 74.89   Mean   :18.124   Mean   :132.15   Mean   :17.541
##  3rd Qu.: 81.87   3rd Qu.:25.613   3rd Qu.:198.97   3rd Qu.:20.298
##  Max.   :142.62   Max.   :42.635   Max.   :393.62   Max.   :25.191
##   ACN.Adjusted     VZ.Adjusted     DIS.Adjusted     CVX.Adjusted
##  Min.   : 29.04   Min.   :14.44   Min.   : 25.54   Min.   : 42.04
##  1st Qu.: 60.81   1st Qu.:29.67   1st Qu.: 49.42   1st Qu.: 70.00
##  Median : 92.96   Median :35.94   Median : 94.34   Median : 81.87
##  Mean   :112.60   Mean   :37.31   Mean   : 88.48   Mean   : 81.63
##  3rd Qu.:152.50   3rd Qu.:46.64   3rd Qu.:109.55   3rd Qu.: 93.94
##  Max.   :334.14   Max.   :57.66   Max.   :201.91   Max.   :110.37
##  INTC.Adjusted    KO.Adjusted     UNH.Adjusted     WFC.Adjusted
##  Min.   :12.52   Min.   :17.26   Min.   : 23.29   Min.   :16.96
##  1st Qu.:18.86   1st Qu.:28.75   1st Qu.: 49.68   1st Qu.:25.51
##  Median :28.08   Median :34.56   Median :108.88   Median :40.38
##  Mean   :31.54   Mean   :35.30   Mean   :145.28   Mean   :36.57
##  3rd Qu.:44.25   3rd Qu.:41.16   3rd Qu.:231.58   3rd Qu.:45.53
##  Max.   :66.43   Max.   :56.18   Max.   :425.56   Max.   :58.15
##   BA.Adjusted     CSCO.Adjusted   CMCSA.Adjusted    MRK.Adjusted
##  Min.   : 46.68   Min.   :10.00   Min.   : 6.686   Min.   :19.86
##  1st Qu.: 64.11   1st Qu.:16.53   1st Qu.:16.665   1st Qu.:31.75
##  Median :122.85   Median :22.97   Median :26.556   Median :44.87
##  Mean   :161.79   Mean   :27.14   Mean   :27.072   Mean   :46.57
##  3rd Qu.:229.73   3rd Qu.:39.03   3rd Qu.:35.726   3rd Qu.:61.15
##  Max.   :430.30   Max.   :58.21   Max.   :58.911   Max.   :81.41
##   PFE.Adjusted     PEP.Adjusted    ORCL.Adjusted     C.Adjusted
##  Min.   : 8.688   Min.   : 43.05   Min.   :18.23   Min.   :19.75
##  1st Qu.:18.613   1st Qu.: 55.70   1st Qu.:28.68   1st Qu.:37.83
##  Median :25.049   Median : 82.49   Median :36.61   Median :44.27
##  Mean   :24.440   Mean   : 85.17   Mean   :39.31   Mean   :46.72
##  3rd Qu.:30.847   3rd Qu.:103.22   3rd Qu.:46.71   3rd Qu.:58.37
##  Max.   :49.604   Max.   :155.74   Max.   :89.95   Max.   :78.08
```

Calculate log-difference

```
logdiff = function(d){
  len = length(d[,1])
  rets =  d[,]
  for(j in 1:length(rets[1,])){
    rets[1:len,j] = diff(log(rets[,j]))
  }
  rets = rets[2:len,]
  return(rets)
}
```

We built factor models based on past data. Firstly, we calculate skewness and kurtosis of historical stock returns in the rolling-basis.

```
Weight_cap = as.numeric(t(as.matrix(stock_weights))*as.matrix(ticker_data[1,]))/(as.matrix(t(stock_weights))%*%t(as.matrix(t
icker_data[1,])))[1,1]
Weight_cap
```

```
##  [1] 0.065741833 0.012434737 0.024860906 0.074914765 0.076316716 0.040270767
##  [7] 0.046556263 0.045538130 0.068564907 0.014090308 0.041850298 0.054742673
## [13] 0.009228002 0.027121697 0.026466934 0.026055699 0.021268772 0.034352543
## [19] 0.026412953 0.031071218 0.009929342 0.038387726 0.012286793 0.030237238
## [25] 0.012965430 0.023225240 0.021902964 0.024993882 0.027460685 0.030750579
```

```
n = length(ticker_data[,1])
Lreturns = ticker_data[,]
Lreturns = logdiff(Lreturns)
skewdata = c()
kurtdata = c()
ret = c()
interval = 63
returns = cumsum(as.xts(Lreturns))
names = c()
for(i in 1:length(symbols)){
  start_window = 1
  end_window = 64
  temp_skewness = c()
  temp_kurtosis = c()
  temp_r = c()
  for(j in 1:(length(Lreturns[,1]) - interval)){ #roll from 1 to the end of the dataset - averaging size
    temp_skewness = rbind(temp_skewness,skewness(Lreturns[(start_window:end_window),i],type = 1)) #rolling Skew calculation
    #type 1 normalizes the measure
    temp_kurtosis = rbind(temp_kurtosis , (kurtosis(Lreturns[(start_window:end_window),i]))) #rolling kurtosis calculation
    r = as.vector(returns[(end_window),i]) - as.vector(returns[(start_window),i]) #individual stock returns

    temp_r = rbind(temp_r, r)

    start_window = start_window + 1
    end_window = end_window + 1
  }
  skewdata = cbind(skewdata,temp_skewness)
  kurtdata = cbind(kurtdata,temp_kurtosis)
  ret = cbind(ret,temp_r)
}
colnames(skewdata) = c(symbols)
colnames(kurtdata) = c(symbols)

average_skewness = rowMeans(skewdata)
reg = Lreturns[((interval + 1):length(Lreturns[,1])),]

ret = ret[((interval + 1):length(ret[,1])),]
average_skewness = average_skewness[1:(length(average_skewness) - interval)]

portfolio_Rets = returns%*%Weight_cap
portfolio_Rets = as.matrix(portfolio_Rets)
row.names(portfolio_Rets) = c(row.names(as.matrix(returns)))
portfolio_Rets = as.xts(portfolio_Rets)

colnames(ret) = c(symbols)
print("portfolio return is :")
```

```
## [1] "portfolio return is :"
```

```
head(portfolio_Rets)
```

```
##                             [,1]
## 2010-04-04 20:00:00  0.0023579023
## 2010-04-05 20:00:00  0.0044600367
## 2010-04-06 20:00:00 -0.0003701026
## 2010-04-07 20:00:00  0.0049920889
## 2010-04-08 20:00:00  0.0121222755
## 2010-04-11 20:00:00  0.0145035584
```

Parameters pre-setting

Use the factor model calculated above with empirical constrains of tracking error and no beta bet to predict the future returns and backtest the result

Calculate the regression with variables skewness and kurtosis in the rolling-window, optimize the quadratic equation by package quadprog which has the optimal solution of portfolio component weights.

```
optimal_port = c()
Benchmark_rets = c()
TE = c()
port_active_weight = c()
port_active_risk = c()
Log_value = c()
Benchmark_i_weight = c()
vol = c()
IR = c()
dates = c()
equally_weights = rep(1/30,30)
equal_weight_returns = c()
t_stat_table_ret = c()
t_stat_table_skew = c()
skew_regress = c()
for(j in 0:30){
  #now we want to predict the next 63 days out

  t = 252*3 + j*63 #tfinal
  ts = 1 + j*63  #t-start


  #these are the times to check for the regression
  #after each iteration we push t-start and t-final out 63 days to our predicted value
  #we then use the new values to calculate a new regression and then predict out
  #another 63 days
  #do this until data is exhausted
  #if done correctly, data should be approximately 28 predictions long, i.e 7 years * 4 quarters
  stock.fit = lm(ret[(ts:(t)),] ~ average_skewness[(ts:(t))])
  t_stat_table_ret = c(t_stat_table_ret,coef(summary(stock.fit)))

  data_nextwindow = average_skewness[t+1]
  skew_regress =rbind(skew_regress,data_nextwindow)

  return_predict = predict(stock.fit,data.frame(data_nextwindow)) #here we predict the return of the stock
  real_volatility = ret^2
  mu = return_predict[1,] #save return for optimization
  kurtregress = kurtdata[(1:(length(kurtdata[,1]) - interval)),]
  skewregress = skewdata[(1:(length(skewdata[,1]) - interval)),]

  volatility_estimate = c()
  for(i in 1:length(symbols)){
    vol.fit = lm(real_volatility[(ts:(t)),i]~ kurtregress[(ts:(t)),i] + abs(skewregress[(ts:t),i]))
    volatility_prediction = predict(vol.fit,newdata = data.frame(kurtregress[(t+1),i],abs(skewregress[(t+1),i])))
    volatility_estimate = cbind(volatility_estimate,volatility_prediction[1])
    t_stat_table_skew = rbind(t_stat_table_skew,coef(summary(vol.fit))[,3])

  }

  #now that we have the estimates of mu and vol we can optimize portfolio
  #caveats:
  # no beta bet, calculate betas for last 3 years

  Benchmark_ret_diff = diff(portfolio_Rets)
  Benchmark_ret_diff = Benchmark_ret_diff[((interval + ts - 1):(t + interval - 1)),]
  Beta = Lreturns[((interval*2+ ts - 1):(t + interval*2 - 1)),]

  beta_cor_m = cor(Beta) #correlation matrix needed for optimization of the portfolio matrix
  #i need a correlation esimate to minimize the estimated covariance matrix
  beta_cov_m = cov(Beta) #base covariance matrix for Tracking Error
  Beta = BetaCoVariance(Beta,Benchmark_ret_diff) #calculated daily beta for last 3 years of data
  volatility_estimate = exp(volatility_estimate)*diag(beta_cov_m)*63

  #no shorts, no leverage, Tracking Error = 0.03 or 0.0009 = Wa'*cov*Wa
  #we can solve using quadratic programming numerically
  #we can use package quadprog to solve the convex function with constraints

  #we can choose what kind of covariance matrix we want to minimize for our optimization program
  #first D designates using estimated variances but uses historical correlations

  D = sqrt(diag(as.numeric(volatility_estimate)))%*%beta_cor_m%*%sqrt(diag(as.numeric(volatility_estimate)))

  #second D designates using the esimated variance but assumes zero correlation between assets
```

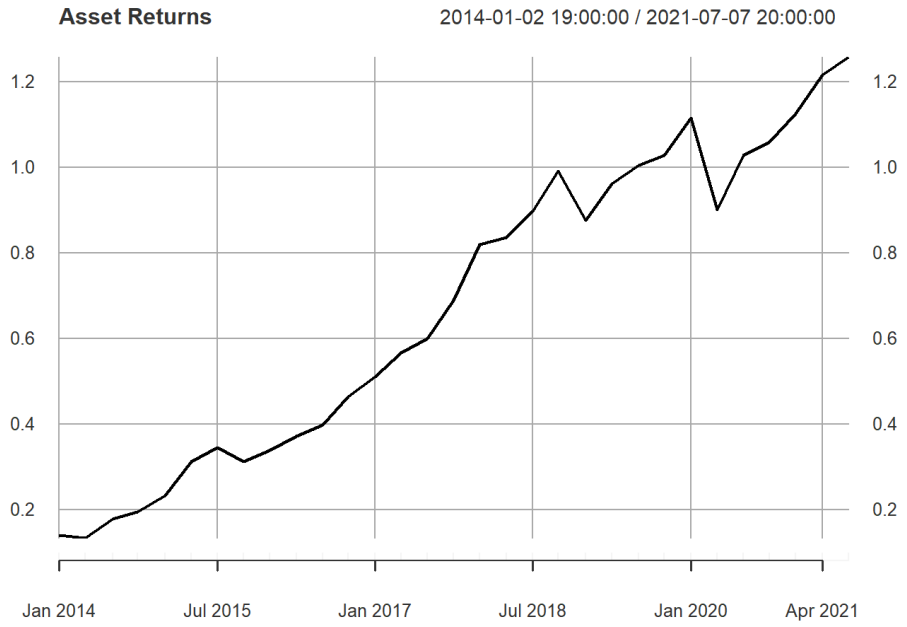
```
  #third D uses just a historical covariance matrix
  A = rep(1,length(symbols))
  ones = rep(1,length(symbols))
  ones = diag(ones)
  zero = rep(0,length(symbols))
  b0 = rbind(1,1)
  b0 = append(b0,zero,after = length(b0))

  A = cbind((A),as.vector(t(Beta)),ones)

  mu = as.matrix(mu)
  L = 0.5

  TE = 1
  x = c()
  equally_weights_after = as.numeric(t(as.matrix(stock_weights))*as.matrix(ticker_data[(t + 1 + 63*2),]))/(as.matrix(t(stock
_weights))%*%t(as.matrix(ticker_data[(t + 1 + 63*2),])))[1,1]
  while(TE > 0.0009){
    if(L > 100000){
      break
    }
    else{
      L = L*2
      W = solve.QP(Dmat = L*D,dvec = (1)*mu,Amat = A, bvec = b0,meq = 2)

      x = W$solution
      TE = t(x - equally_weights_after)%*%(beta_cov_m*63)%*%(x-equally_weights_after)

    }
  }
  port_active_weight = rbind(port_active_weight,x)


  optimal_port = rbind(optimal_port,t(x)%*%ret[(t+64),])
  equal_weight_returns = rbind(equal_weight_returns,t(equally_weights)%*%ret[(t+64),])
  Benchmark_rets = rbind(Benchmark_rets,as.numeric(portfolio_Rets[(t + 1 + 63*3)]) - as.numeric(portfolio_Rets[(t + 1 + 63*2
)]))
  dates = rbind(dates,as.character(index(portfolio_Rets[(t+1+63*3)])))

  Benchmark_i_weight = rbind(Benchmark_i_weight, equally_weights_after)
  #row.names(Benchmark_i_weight)[i,] = c()

  port_active_risk = rbind(port_active_risk,TE)
  Log_value = rbind(Log_value,L)
  IR = rbind(IR,sqrt(TE)*2*L)


}
```

Calculate the criteria we are interested in, which are IR and IC, and plot the returns of each portfolio and benchmark.

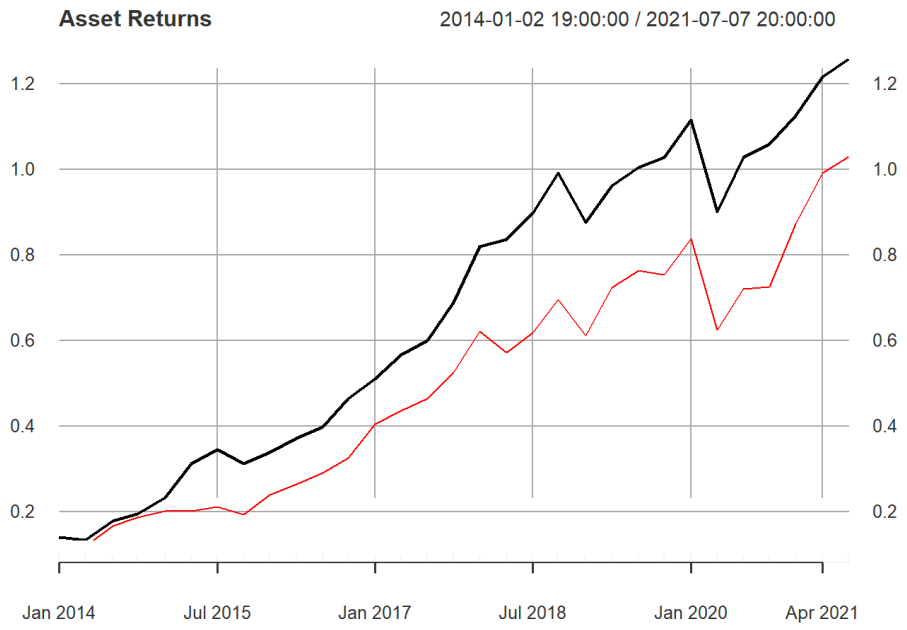
```
IC = IR/sqrt(4)
colnames(Benchmark_i_weight) = c(symbols)
colnames(port_active_weight) = c(symbols)
rownames(optimal_port) = c(dates)
rownames(Benchmark_rets) = c(dates)
rownames(equal_weight_returns) = c(dates)
#fix out of bounds error
post_risk = sqrt(var(optimal_port - Benchmark_rets))*sqrt(4)
post_alpha = mean(optimal_port - Benchmark_rets)*4
sqrt(port_active_risk)
```

```
##             [,1]
##  [1,] 0.02891466
##  [2,] 0.02218882
##  [3,] 0.02406993
##  [4,] 0.01876248
##  [5,] 0.01960557
##  [6,] 0.02885161
##  [7,] 0.02011803
##  [8,] 0.01928183
##  [9,] 0.02666875
## [10,] 0.02644584
## [11,] 0.02712799
## [12,] 0.02487374
## [13,] 0.02725908
## [14,] 0.02903508
## [15,] 0.01995760
## [16,] 0.02558449
## [17,] 0.02112535
## [18,] 0.02357626
## [19,] 0.01976592
## [20,] 0.01934935
## [21,] 0.02168741
## [22,] 0.02782773
## [23,] 0.02401844
## [24,] 0.02513986
## [25,] 0.02338888
## [26,] 0.02683156
## [27,] 0.03126549
## [28,] 0.02808465
## [29,] 0.02262821
## [30,] 0.02332294
## [31,] 0.03349032
```

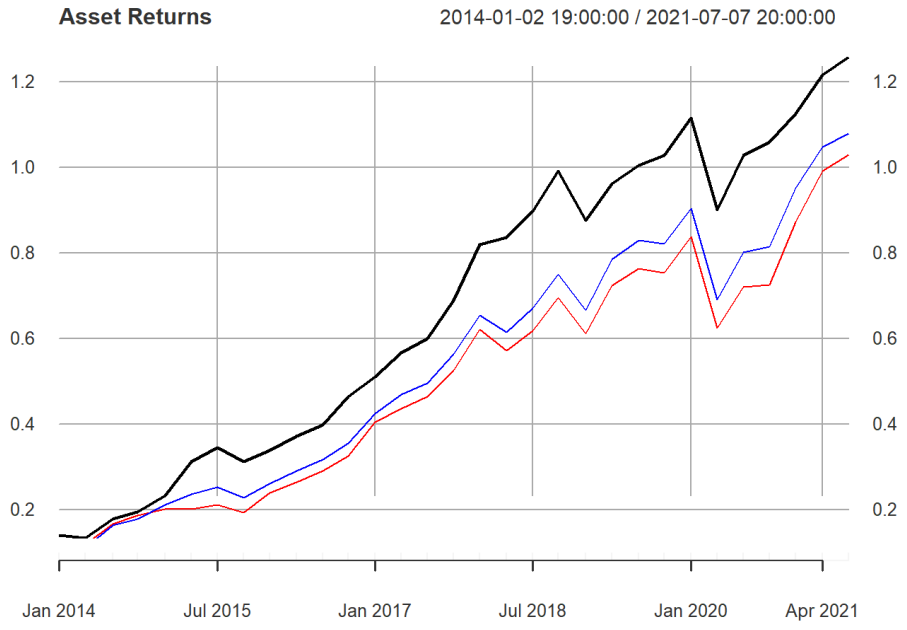
```
plot(cumsum(as.xts(optimal_port)),type = "l",main = "Asset Returns")
```




```
lines(cumsum(as.xts(Benchmark_rets)),type = "l",col ="red")
```




```
lines(cumsum(as.xts(equal_weight_returns)),type = "l",col = "blue")
```




```
post_IR = post_alpha/post_risk
post_IC = post_IR/sqrt(4)
```

Print the final statistical result of each factor.

```
exante_table1 = cbind(sqrt(port_active_risk),IR,IC) #ex-ante
posthoc_table2 = cbind(post_risk,post_IR,post_IC,post_alpha) #post-hoc
colnames(posthoc_table2) = c("post-TE","post-IR","post-IC","post-alpha")
colnames(exante_table1) = c("ex-ante-TE","ex-ante-IR","ex-ante-IC")
#write.table(exante_table1,file = "ex_ante.csv",sep = ",",col.names = TRUE)
#write.table(posthoc_table2,file = "post_hoc.csv",sep = ",",col.names = TRUE)
t_stat_table_skew = as.matrix(t_stat_table_skew)

#algo to extract t-stat averages and variances
write.table(t_stat_table_ret,file = "ret.csv",sep = ",")
t_stats_table = read.csv("ret.csv",header = TRUE,fill = FALSE)
t_stats_table = t(t_stats_table)
simulate_t_table = c()
for(i in (1:length(t_stats_table[,1]) + 1)){
  if(i%%4 == 0){
    simulate_t_table = rbind(simulate_t_table,t_stats_table[i-1,])
  }
}
print("Mean coefficient of regression on skewness")
```

```
## [1] "Mean coefficient of regression on skewness"
```

```
colMeans(simulate_t_table)
```

```
##                (Intercept) average_skewness[(ts:(t))]
##                  11.630440                 -2.752275
```

```
simulate_t_tablet = c()
for(i in (1:length(t_stat_table_skew[,1]) + 1)){
  if(i%%4 == 0){
    simulate_t_tablet = rbind(simulate_t_tablet,t_stat_table_skew[i-1,])
  }
}

print("Mean coefficient of regression on skewness and kurtosis")
```

```
## [1] "Mean coefficient of regression on skewness and kurtosis"
```

```
colMeans(simulate_t_tablet)
```

```
##                (Intercept)    kurtregress[(ts:(t)), i]
##                 14.65020457                -0.47197276
## abs(skewregress[(ts:t), i])
##                  0.09238809
```

It is clear that the skewness and kurtosis have negative impact on the future stock returns, which matches the empitical studies towards the impact of skewness and kurtosis to the stock returns.

References

https://www.tidytextmining.com/sentiment.html#
(https://www.tidytextmining.com/sentiment.html#):~:text=The%20AFINN%20lexicon%20assigns%20words,positive%20scores%20indicating%20positive%20se

Eric Jondeau, Qunzi Zhang, Xiaoneng Zhu, Average Skewness Matters, Journal of Financial Economics, Volume 134, issue 1, 2019, pages 29-47, ISSN 0304-405X, https://doi.org/10.1016/j.jfineco.2019.03.003 (https://doi.org/10.1016/j.jfineco.2019.03.003).

Youngmin Choi, Suzanne S. Lee, Realized Skewness and Future Stock Returns: The Role of Information, 2004

https://www.kaggle.com/datasets/therohk/india-headlines-news-dataset (https://www.kaggle.com/datasets/therohk/india-headlines-news-dataset)

https://www.rdocumentation.org/packages/quadprog/versions/1.5-8/topics/solve.QP
(https://www.rdocumentation.org/packages/quadprog/versions/1.5-8/topics/solve.QP)

https://www.sciencedirect.com/science/article/pii/S0304405X15001257 (https://www.sciencedirect.com/science/article/pii/S0304405X15001257)

https://www.sciencedirect.com/topics/social-sciences/skewness (https://www.sciencedirect.com/topics/social-sciences/skewness)

https://www.investopedia.com/terms/k/kurtosis.asp (https://www.investopedia.com/terms/k/kurtosis.asp)