Final Project

Mingze Li 300137754

2023-12-23

1 Introduction

Stock plays a crucial role in the financial market. Stock provides a convenient capital formation, income divide, and ownership divide vehicle. Public trading stocks are also a very popular investment option. This project aims to use data mining tools to model and analyze the stock market, then try to get stock price prediction and investment suggestions based on the stock price data.

The dataset "sep 100" contains 100 representative firms' stock prices from different industries from 2000 to 2023. There are 5786 records with 101 variables. The first column is character type data that indicates dates, and all others are double type data that indicates stock price. But some firms' prices are not available for the entire period(2000~2023) under study. Therefore we have to remove firms with missing values. The code below displays all firms that do not have missing values.

Note that some R versions cannot knit the code "library(ClusterR)", "library(StatMatch)", and "library(dplyr)". So the non-comment line is in a hidden chunk. To read the data. Please place the file "sep100.csv" and this RMD file into the same folder.

```
library(ggplot2)
#library(StatMatch)
library("ClusterR")
library(cluster)
#library(dplyr)
library(timeDate)
library(factoextra)
library(readr)
#library(lubridate)
library(TTR)  # For EMA and other technical indicators
#library(zoo)  # For rollmean and rollapply
library(ggplot2)
library(scales)
library(keras)
library(corrplot)
data <- read.csv("sep100.csv",header = TRUE)
head(data)
```

```
##                  Date      AAPL       VZ      WBA      WFC      WMT      XOM
## 1 2000-07-13 00:00:00 1.008929 46.31512 30.93750 21.40625 59.12500 39.00000
## 2 2000-07-14 00:00:00 1.030134 44.51648 31.25000 21.84375 59.50000 38.78125
## 3 2000-07-17 00:00:00 1.041295 44.51648 31.21875 21.68750 60.71875 39.37500
## 4 2000-07-18 00:00:00 1.022321 43.89819 31.18750 21.03125 60.06250 39.09375
## 5 2000-07-19 00:00:00 0.940848 42.26817 31.31250 21.09375 60.00000 39.34375
```

```
## 6 2000-07-20 00:00:00 0.984375 39.79503 31.62500 21.65625 60.00000 39.06250
##        ABT     ADBE      AIG     AMGN    AMT     AMZN AVGO      AXP       BA
## 1 19.58330 34.46875 1603.333 69.43750 46.500 1.750000   NA 48.19621 44.56250
## 2 19.07829 34.04688 1598.333 70.75000 45.750 2.131250   NA 49.50916 44.43750
## 3 19.07829 33.76562 1591.667 72.37500 44.125 2.056250   NA 49.29033 43.96875
## 4 18.82578 32.73438 1589.167 74.85938 44.500 2.087500   NA 48.30562 45.12500
## 5 18.46105 32.48438 1590.000 73.43750 44.500 2.043750   NA 48.19621 45.87500
## 6 18.46105 34.31250 1625.000 74.06250 45.125 2.015625   NA 49.78269 46.06250
##        BAC     BIIB       BK    BKNG     BLK      BMY BRK.B        C      CAT
## 1 23.09375 38.12500 49.15730 230.625 33.6875 51.39596    NA 499.6875 17.96875
## 2 23.75000 39.33333 49.75355 236.625 33.8750 52.22876    NA 510.0000 17.87500
## 3 23.43750 42.27083 51.31042 237.375 33.0625 52.39235    NA 497.8125 17.85938
## 4 23.12500 41.41667 51.07854 232.500 32.3125 50.50367    NA 500.1562 18.87500
## 5 23.09375 39.83333 51.07854 238.500 32.1250 49.84932    NA 510.0000 18.96875
## 6 24.18750 40.04167 52.27104 243.375 32.5625 48.52576    NA 529.2188 18.53125
##   CHTR       CL    CMCSA      COF      COP    COST CRM    CSCO      CVS
## 1   NA 26.53125 12.22917 51.31250 19.48672 36.5000  NA 65.2500 21.46875
## 2   NA 27.12500 12.29167 51.00000 19.53436 36.0000  NA 68.2500 21.71875
## 3   NA 27.46875 12.35417 50.21875 19.43907 35.5000  NA 69.6250 22.25000
## 4   NA 27.46875 12.00000 52.43750 19.53436 33.8750  NA 67.2500 22.00000
## 5   NA 27.25000 11.91667 53.25000 19.65347 32.9375  NA 66.8125 21.25000
## 6   NA 27.68750 12.08333 55.50000 19.48672 32.9375  NA 69.5000 21.68750
##        CVX       DD      DHR      DIS DOW      DUK      EMR      EXC        F
## 1 41.97656 47.63506 8.801649 36.62381  NA 53.44304 31.81250 15.33523 27.56446
## 2 41.87500 45.18882 8.671342 36.93209  NA 52.84378 31.75000 15.22379 27.17119
## 3 41.33594 44.07689 8.635804 36.06890  NA 53.08893 30.82812 15.62500 26.50979
## 4 40.71875 43.89898 8.837187 35.26738  NA 54.09678 30.67188 15.73645 26.84943
## 5 40.96875 43.81002 8.801649 36.50050  NA 55.84008 30.79688 15.89248 27.13544
## 6 40.00000 42.34227 9.050417 38.10356  NA 54.42365 32.35938 15.51355 27.09969
##   FB     FDX       GD       GE     GILD GM GOOG GOOGL       GS      HD      HON
## 1 NA 41.8125 26.71875 315.2585 2.568359 NA   NA    NA 104.2500 56.1875 34.14255
## 2 NA 42.4375 26.81250 309.2536 2.519531 NA   NA    NA 104.1875 56.4375 34.44048
## 3 NA 40.9375 27.18750 322.3894 2.578125 NA   NA    NA 100.0000 56.8750 35.98970
## 4 NA 40.9375 26.53125 313.7573 2.505859 NA   NA    NA 103.8750 57.3750 35.15550
## 5 NA 40.0000 27.06250 316.7597 2.335938 NA   NA    NA 100.7500 57.1250 35.45343
## 6 NA 39.7500 28.93750 326.1424 2.265625 NA   NA    NA 103.6875 58.0625 34.55965
##         IBM     INTC      JNJ      JPM KHC       KO      LIN       LLY     LMT
## 1  99.42638 71.59375 47.62500 50.75000  NA 28.96875 19.65625  95.75000 25.0000
## 2  99.36664 73.34375 45.90625 52.50000  NA 28.81250 19.56250  94.50000 26.3750
## 3 100.86042 73.15625 47.17188 52.03125  NA 28.67188 19.28125  98.46875 25.3750
## 4  98.76912 71.50000 47.50000 50.93750  NA 29.09375 19.25000 100.50000 25.3125
## 5 103.96750 69.06250 46.59375 50.68750  NA 30.25000 19.31250  98.75000 25.3750
## 6 112.09369 71.34375 45.93750 51.31250  NA 30.25000 19.68750  96.50000 26.7500
##        LOW MA     MCD MDLZ     MDT      MET      MMM      MO      MRK     MS
## 1 11.87500 NA 31.5000   NA 50.7500 17.65820 44.78125 24.8750 65.95897 96.375
## 2 11.96875 NA 31.3750   NA 50.2500 18.60517 43.93750 24.5000 64.16985 96.000
## 3 12.00000 NA 31.3125   NA 50.8750 18.10383 43.62500 23.6250 64.28912 92.125
## 4 12.03125 NA 31.3125   NA 52.0625 18.15954 44.00000 24.0625 63.33492 91.250
## 5 11.70312 NA 31.3750   NA 50.5000 18.21524 43.25000 25.0000 62.26145 88.250
## 6 11.56250 NA 31.6875   NA 51.4375 18.32665 43.25000 25.5000 61.12834 91.250
##       MSFT      NEE NFLX      NKE     NVDA     ORCL     PEP      PFE       PG
## 1 39.96875 6.765625   NA 5.492188 3.098958 37.87500 40.3750 43.76186 27.28125
## 2 39.46875 6.765625   NA 5.484375 3.075521 38.06250 41.6875 42.75380 28.00000
## 3 39.09375 6.730469   NA 5.652344 3.041667 38.06250 42.8750 43.74704 28.06250
```

```
## 4 39.25000 6.601563   NA 5.617188 2.981771 37.09375 43.0000 42.57590 29.12500
## 5 36.56250 6.687500   NA 5.507813 2.791667 36.87500 43.3750 44.05835 29.56250
## 6 37.40625 6.648438   NA 5.578125 2.867188 39.06250 43.7500 42.69450 29.50000
##   PM PYPL     QCOM      RTX     SBUX       SO      SPG        T      TGT
## 1 NA   NA 31.00000 19.03713 5.000000 14.92887 23.20358 34.55438 31.06250
## 2 NA   NA 31.53125 19.23380 5.000000 14.73796 23.20358 34.27115 31.06250
## 3 NA   NA 34.90625 19.04696 4.976563 14.73796 22.97154 32.68977 30.45312
## 4 NA   NA 32.56250 18.56513 4.968750 14.85251 22.97154 32.85498 29.65625
## 5 NA   NA 31.50000 18.09314 4.968750 15.34886 23.20358 32.94939 29.53125
## 6 NA   NA 31.53125 18.05381 5.000000 15.34886 23.20358 32.57175 30.50000
##        TMO TMUS TSLA     TXN      UNH      UNP     UPS     USB  V ABBV ACN
## 1 23.12500   NA   NA 70.8750 10.71875 10.25000 60.3125 21.5625 NA   NA  NA
## 2 23.37500   NA   NA 72.5000 10.56250 10.87500 60.8750 21.9375 NA   NA  NA
## 3 22.90625   NA   NA 73.5000 10.86719 11.01562 60.4375 21.4375 NA   NA  NA
## 4 23.31250   NA   NA 70.5625 10.40625 10.98438 60.7500 20.8750 NA   NA  NA
## 5 22.93750   NA   NA 67.2500 10.49219 10.57812 61.2500 21.0625 NA   NA  NA
## 6 23.25000   NA   NA 67.7500 10.60938 10.48438 59.5625 21.3750 NA   NA  NA
```

```r
df <- as.data.frame(
  cbind(
    lapply(
      lapply(data, is.na), sum)
    )
  )
rownames(subset(df, df$V1 ==0))
```

```
##  [1] "Date"  "AAPL"  "VZ"    "WBA"   "WFC"   "WMT"   "XOM"   "ABT"   "ADBE"
## [10] "AIG"   "AMGN"  "AMT"   "AMZN"  "AXP"   "BA"    "BAC"   "BIIB"  "BK"
## [19] "BKNG"  "BLK"   "BMY"   "C"     "CAT"   "CL"    "CMCSA" "COF"   "COP"
## [28] "COST"  "CSCO"  "CVS"   "CVX"   "DD"    "DHR"   "DIS"   "DUK"   "EMR"
## [37] "EXC"   "F"     "FDX"   "GD"    "GE"    "GILD"  "GS"    "HD"    "HON"
## [46] "IBM"   "INTC"  "JNJ"   "JPM"   "KO"    "LIN"   "LLY"   "LMT"   "LOW"
## [55] "MCD"   "MDT"   "MET"   "MMM"   "MO"    "MRK"   "MS"    "MSFT"  "NEE"
## [64] "NKE"   "NVDA"  "ORCL"  "PEP"   "PFE"   "PG"    "QCOM"  "RTX"   "SBUX"
## [73] "SO"    "SPG"   "T"     "TGT"   "TMO"   "TXN"   "UNH"   "UNP"   "UPS"
## [82] "USB"
```

The data_cleaned is the data set without missing values. It contains 81 firms, and 5786 days closing stock price. This is a very high dimensional dataset and contains large numbers of observations. Therefore it is necessary to conduct dimension reduction to pick some firms that have the most representative features. To reduce the complexity of the dataset.

```r
data_cleaned <- data[, cbind(rownames(subset(df, df$V1 ==0)))]
#head(data_cleaned) ## Companies with missing data are gone.
data=c()
```

2 Dimension reduction

2.1 Most Representative Firms

In order to make this data set fit better for supervised and unsupervised learning models, it is necessary to conduct dimension reduction to extract sufficient features for model for training or reducing the complexity of the dataset.

The first method we conducted is PAM with Gower's distance. This method would group the similar firms together, and then highlight the most representative firms in terms of similarity in each cluster. In order to group firms in terms of behavior, we need to define a new dataset containing the "movement" of the stock price. Daily loss is a good measure that can present this movement, its definition is:

$$dailyloss = yesterday's\, price - todays'\, price$$

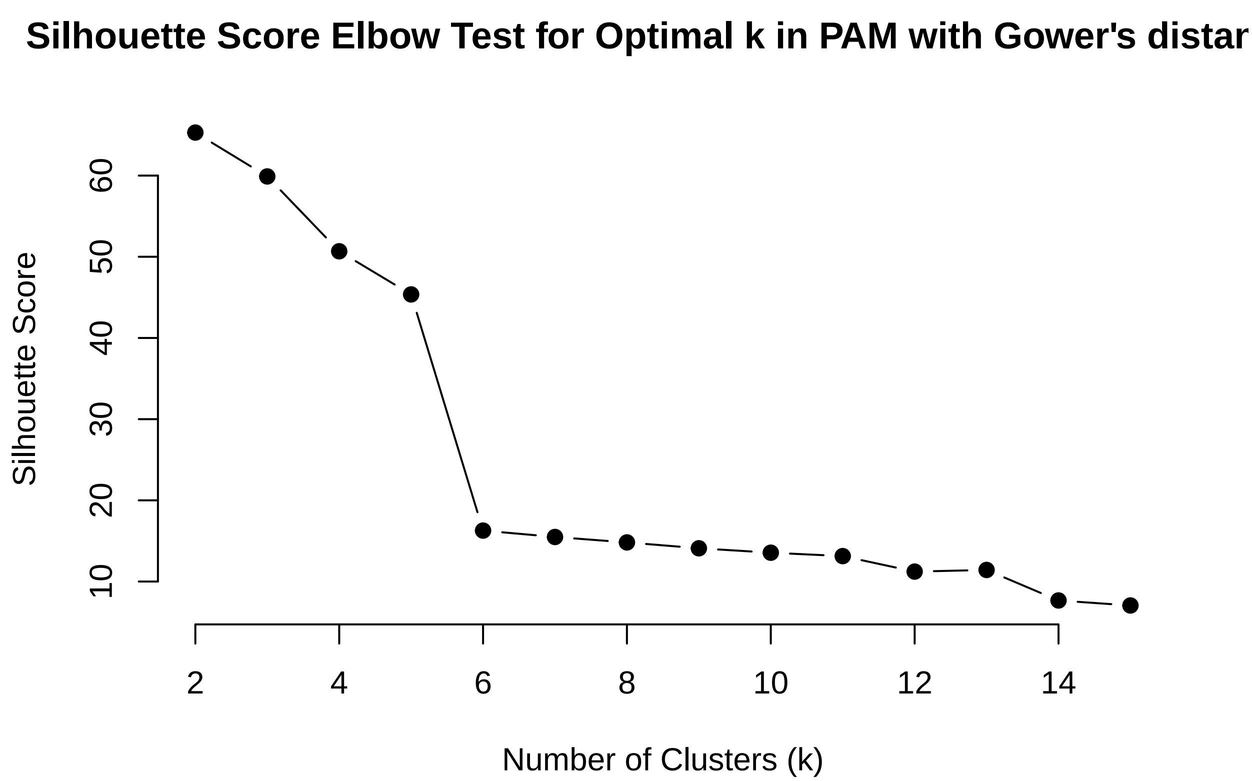
```
data_loss=data_cleaned

diff=0
for (i in 2:length(data_cleaned)) {
    for (j in 2:length(data_cleaned[,1])) {
      diff=data_cleaned[j-1,i]-data_cleaned[j,i]
      data_loss[j-1,i]=diff

    }
  data_loss[length(data_cleaned[,1]),i]=NA

}
data_loss=na.omit(data_loss)

head(data_loss)##The loss data.
```

```
##                  Date        AAPL         VZ       WBA      WFC      WMT
## 1 2000-07-13 00:00:00 -0.02120495  1.7986450 -0.31250 -0.43750 -0.37500
## 2 2000-07-14 00:00:00 -0.01116109  0.0000000  0.03125  0.15625 -1.21875
## 3 2000-07-17 00:00:00  0.01897407  0.6182823  0.03125  0.65625  0.65625
## 4 2000-07-18 00:00:00  0.08147299  1.6300240 -0.12500 -0.06250  0.06250
## 5 2000-07-19 00:00:00 -0.04352701  2.4731369 -0.31250 -0.56250  0.00000
## 6 2000-07-20 00:00:00  0.02790201 -3.3724594 -0.37500  0.46875  0.31250
##        XOM       ABT      ADBE        AIG      AMGN    AMT        AMZN
## 1  0.21875 0.5050144  0.421875   5.000000 -1.312500  0.750 -0.38124990
## 2 -0.59375 0.0000000  0.281250   6.666748 -1.625000  1.625  0.07499981
## 3  0.28125 0.2525063  1.031250   2.500000 -2.484375 -0.375 -0.03125000
## 4 -0.25000 0.3647327  0.250000  -0.833374  1.421875  0.000  0.04375005
## 5  0.28125 0.0000000 -1.828125 -35.000000 -0.625000 -0.625  0.02812505
## 6  0.50000 0.3366756  0.640625  49.166626 -3.937500  0.500 -0.04062510
##          AXP       BA      BAC       BIIB         BK   BKNG     BLK        BMY
## 1 -1.3129501  0.12500 -0.65625 -1.2083321 -0.5962486 -6.000 -0.1875 -0.8328056
## 2  0.2188225  0.46875  0.31250 -2.9375000 -1.5568695 -0.750  0.8125 -0.1635857
## 3  0.9847145 -1.15625  0.31250  0.8541641  0.2318764  4.875  0.7500  1.8886833
## 4  0.1094131 -0.75000  0.03125  1.5833359  0.0000000 -6.000  0.1875  0.6543465
## 5 -1.5864830 -0.18750 -1.09375 -0.2083359 -1.1924973 -4.875 -0.4375  1.3235626
## 6 -2.0788383 -0.75000 -0.18750  0.2708359 -0.7287445  3.000 -1.2500  0.8774223
##           C       CAT       CL       CMCSA      COF         COP   COST    CSCO
## 1 -10.31250  0.093750 -0.59375 -0.06250000  0.31250 -0.04764557 0.5000 -3.0000
## 2  12.18750  0.015625 -0.34375 -0.06250000  0.78125  0.09529114 0.5000 -1.3750
## 3  -2.34375 -1.015625  0.00000  0.35416698 -2.21875 -0.09529114 1.6250  2.3750
## 4  -9.84375 -0.093750  0.21875  0.08333302 -0.81250 -0.11911201 0.9375  0.4375
## 5 -19.21875  0.437500 -0.43750 -0.16666603 -2.25000  0.16675758 0.0000 -2.6875
## 6  -2.34375  0.281250 -0.62500  0.64583302 -0.31250  0.47644806 0.6875  1.3750
##        CVS       CVX         DD        DHR        DIS        DUK       EMR
## 1 -0.25000  0.1015625  2.44624329  0.13030720 -0.3082809  0.5992622  0.062500
## 2 -0.53125  0.5390625  1.11193085  0.03553772  0.8631897 -0.2451515  0.921875
```

```
## 3  0.25000  0.6171875  0.17790985 -0.20138264  0.8015289 -1.0078468  0.156250
## 4  0.75000 -0.2500000  0.08895493  0.03553772 -1.2331238 -1.7432976 -0.125000
## 5 -0.43750  0.9687500  1.46774673 -0.24876785 -1.6030617  1.4164276 -1.562500
## 6  0.15625  0.7500000 -0.35581970  0.08292294  1.1714668  0.1634369 -0.203125
##          EXC           F     FDX       GD         GE        GILD      GS
## 1  0.1114473  0.39326859 -0.6250 -0.09375   6.004913  0.04882789  0.0625
## 2 -0.4012127  0.66140366  1.5000 -0.37500 -13.135773 -0.05859399  4.1875
## 3 -0.1114483 -0.33963966  0.0000  0.65625   8.632080  0.07226610 -3.8750
## 4 -0.1560268 -0.28601265  0.9375 -0.53125  -3.002441  0.16992092  3.1250
## 5  0.3789234  0.03575134  0.2500 -1.87500  -9.382690  0.07031298 -2.9375
## 6 -0.1783171  0.00000000 -0.3125 -0.03125   1.125916 -0.08007789 -0.5625
##        HD        HON         IBM     INTC       JNJ       JPM         KO
## 1 -0.2500 -0.2979279  0.05974579 -1.75000  1.718750 -1.750000  0.1562500
## 2 -0.4375 -1.5492249 -1.49378204  0.18750 -1.265625  0.468750  0.1406250
## 3 -0.5000  0.8341980  2.09130096  1.65625 -0.328125  1.093750 -0.4218750
## 4  0.2500 -0.2979279 -5.19837952  2.43750  0.906250  0.250000 -1.1562500
## 5 -0.9375  0.8937836 -8.12619019 -2.28125  0.656250 -0.625000  0.0000000
## 6  0.9375  1.4300537  2.39005280  2.25000 -0.187500 -1.421875 -0.1796875
##        LIN      LLY     LMT       LOW     MCD     MDT         MET      MMM
## 1  0.09375  1.25000 -1.3750 -0.093750  0.1250  0.5000 -0.94696999  0.84375
## 2  0.28125 -3.96875  1.0000 -0.031250  0.0625 -0.6250  0.50133705  0.31250
## 3  0.03125 -2.03125  0.0625 -0.031250  0.0000 -1.1875 -0.05570412 -0.37500
## 4 -0.06250  1.75000 -0.0625  0.328125 -0.0625  1.5625 -0.05570412  0.75000
## 5 -0.37500  2.25000 -1.3750  0.140625 -0.3125 -0.9375 -0.11140823  0.00000
## 6 -0.18750 -1.37500  0.4375  0.281250  0.8125  0.9375 -0.16711235  0.06250
##        MO        MRK      MS     MSFT         NEE          NKE        NVDA
## 1  0.3750  1.7891235  0.3750  0.50000  0.00000000  0.007812977  0.02343702
## 2  0.8750 -0.1192780  3.8750  0.37500  0.03515577 -0.167969227  0.03385401
## 3 -0.4375  0.9542007  0.8750 -0.15625  0.12890625  0.035156250  0.05989599
## 4 -0.9375  1.0734711  3.0000  2.68750 -0.08593702  0.109375000  0.19010401
## 5 -0.5000  1.1331100 -3.0000 -0.84375  0.03906202 -0.070312023 -0.07552099
## 6  0.2500  0.2981873 -3.4375  1.25000  0.05468798 -0.007812977 -0.03645802
##       ORCL     PEP        PFE       PG     QCOM         RTX        SBUX
## 1 -0.18750 -1.3125  1.0080643 -0.71875 -0.53125 -0.19666481  0.000000000
## 2  0.00000 -1.1875 -0.9932404 -0.06250 -3.37500  0.18683243  0.023437023
## 3  0.96875 -0.1250  1.1711349 -1.06250  2.34375  0.48182869  0.007812977
## 4  0.21875 -0.3750 -1.4824486 -0.43750  1.06250  0.47199440  0.000000000
## 5 -2.18750 -0.3750  1.3638535  0.06250 -0.03125  0.03933334 -0.031250000
## 6  1.34375 -0.5000 -0.8301735 -0.06250  0.12500 -0.43266296  0.007812023
##           SO         SPG           T       TGT      TMO     TXN         UNH
## 1  0.1909065  0.00000000  0.28323364  0.000000 -0.25000 -1.6250  0.15625000
## 2  0.0000000  0.23203659  1.58138275  0.609375  0.46875 -1.0000 -0.30468845
## 3 -0.1145430  0.00000000 -0.16521835  0.796875 -0.40625  2.9375  0.46093845
## 4 -0.4963570 -0.23203659 -0.09440994  0.125000  0.37500  3.3125 -0.08593845
## 5  0.0000000  0.00000000  0.37764359 -0.968750 -0.31250 -0.5000 -0.11718655
## 6 -0.0381813 -0.05800819  0.09440994  0.562500  0.81250  4.2500 -0.01562500
##         UNP     UPS     USB
## 1 -0.625000 -0.5625 -0.3750
## 2 -0.140625  0.4375  0.5000
## 3  0.031250 -0.3125  0.5625
## 4  0.406250 -0.5000 -0.1875
## 5  0.093750  1.6875 -0.3125
## 6  0.078125  0.0625  0.4375
```

Manipulate the data set, remove the date column, and put the date as column and firms as rows, then apply the Silhouette Score test to find the optimal cluster number. Silhouette Score is the sum scaled sum of the distance between each cluster member to its cluster centers. In the graph, six is the point that owns the last large drop of Silhouette Score. Therefore the optimal k should be 6.

```
data_losst=t((data_loss[,2:82]))
dist=as.dist(gower.dist(data_losst))

silhouette_scores_pam <- function(k) {
  pam_model <- pam(dist, k = k,metric="manhattan")
  return(sum(silhouette(pam_model$clustering, dist)[,3]))
}

# Apply the function for a range of k values
k_values <- 2:15  # Silhouette score is typically calculated for k >= 2
silhouette_values <- sapply(k_values, silhouette_scores_pam)

# Plot the Silhouette Score
plot(k_values, silhouette_values, type="b", pch = 19, frame=FALSE,
     xlab="Number of Clusters (k)", ylab="Silhouette Score",
     main="Silhouette Score Elbow Test for Optimal k in PAM with Gower's distance")
```



Then apply the PAM method. There are 54 firms in the first cluster, and 23 in the second. All other clusters only have one firm. Then take the medoid and medoid ID(in the original cleaned dataset) for future study.

```
a=pam(dist,6,metric="manhattan")

table(a$clustering)
```

```
##
##  1  2  3  4  5  6
## 54 23  1  1  1  1
```

```
a$medoids
```

```
## [1] "CMCSA" "RTX"   "AIG"   "BKNG"  "BLK"   "C"
```

```
a$id.med+1
```

```
## [1] 25 71 10 19 20 22
```

```
data_losst=c()
```

2.2 Most Volatile Firms

PCA could build a linear combination of firms to principle combinations, and order them in a descending order in terms of variance. It reduces the dimensionality of the data by transforming it into a new coordinate system, where the greatest variance lies on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. It would work well in finding the most volatile firms. PCA algorithm only allows numerical input, so first, remove the date column.

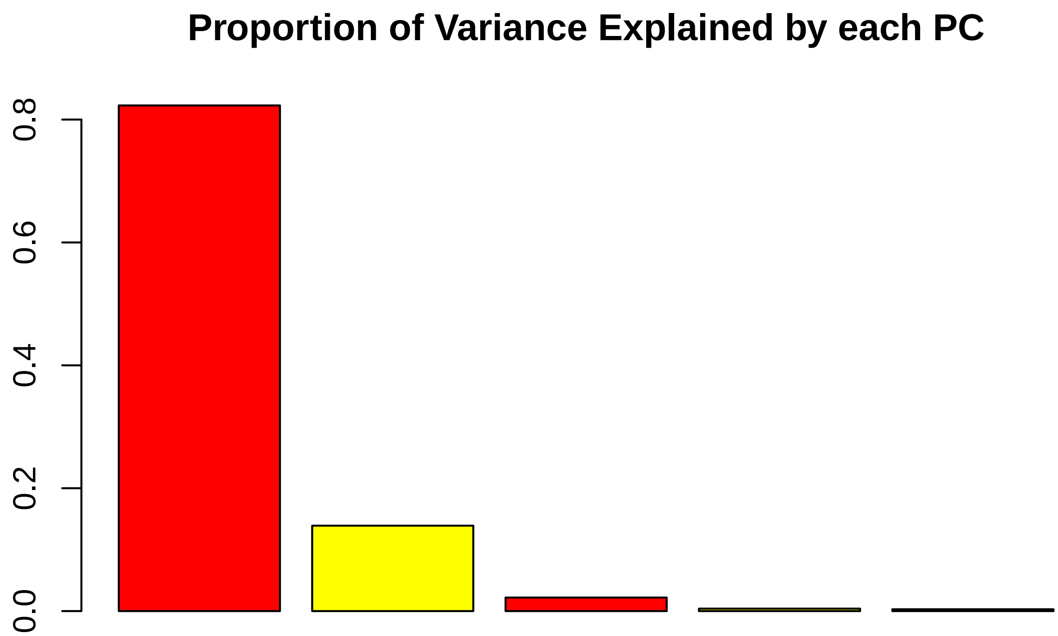
```
data_time <- data_cleaned$Date
data_cleaned_without_time <- data_cleaned[, cbind(rownames(subset(df, df$V1 ==
0)))][,-1]
#head(data_cleaned_without_time)
```

After obtaining the data without the "Date" column, perform PAM to obtain a reduced dataset that contains the most significance in the original dataset.

```
PCA=prcomp(data_cleaned_without_time)
#summary(PCA)
(proportionOfVariance=round(PCA$sdev^2/sum(PCA$sdev^2),3))
```

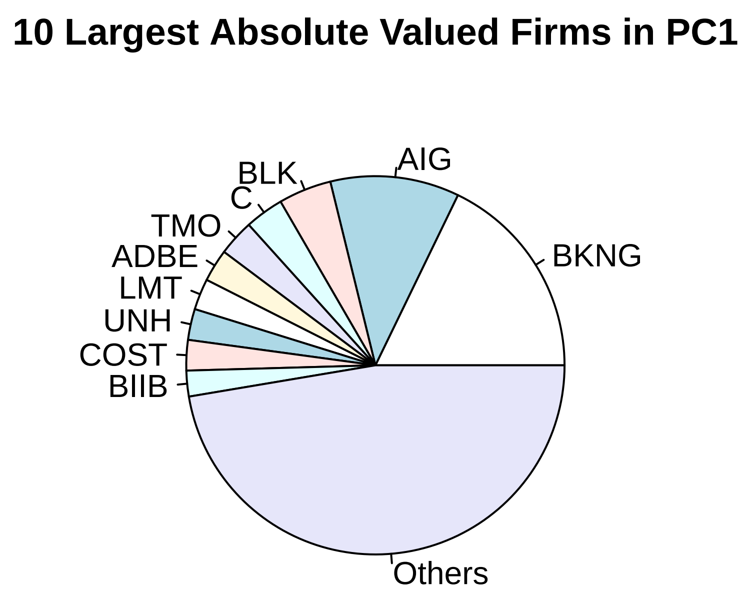
```
##  [1] 0.823 0.139 0.022 0.004 0.003 0.002 0.002 0.001 0.001 0.001 0.000 0.000
## [13] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
## [25] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
## [37] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
## [49] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
## [61] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
## [73] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

```
barplot(proportionOfVariance[1:5], main = "Proportion of Variance Explained by each PC", col = c("red",
```



The graph shows that PC1 contains 82.3% of the original dataset, which could be considered the major source of volatility.

```
order=order(abs(PCA$rotation[,1]),decreasing=T)
selected_data <- c(abs(PCA$rotation[,1])[order[1:10]],sum(abs(PCA$rotation[,1])[order[11:length(order)]]
names(selected_data)[11]="Others"
pie(selected_data,main="10 Largest Absolute Valued Firms in PC1")
```



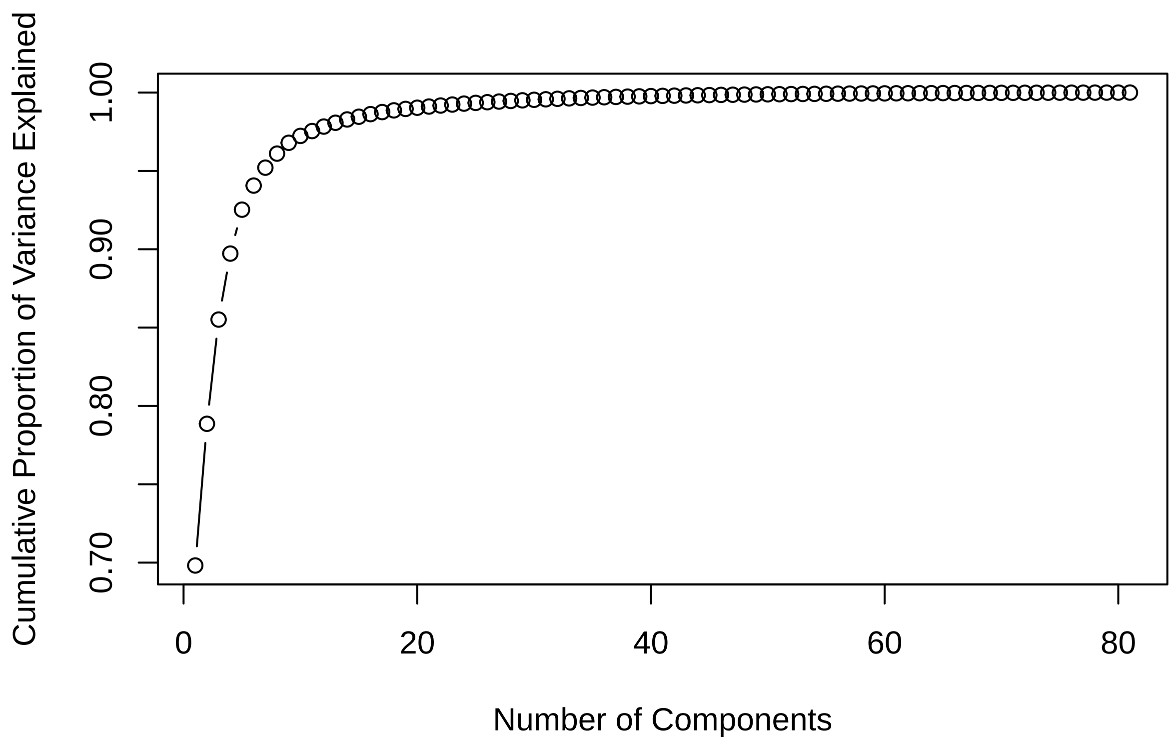
The absolute value in principle component 1 for each firm is the multiple of that firm. A firm with the largest multiple would relatively dominate PC1 more. Therefore contributes more to the variance of the entire data. Thus, AIGNG and AIG should be the two most volatile firms. We found that AIG and AIGNG are the most volatile firms. Note that they are also the only firm clusters in 2.1. Another only firm cluster is C, which is considered the fourth most volatile firm.

2.3 Formal PCA analysis

Used the proportion of explained variance to filter the principal components, To retain a sufficient amount of information while reducing the dimensionality of the dataset. It can reduce the dimensionality of the dataset, address noise, and enhance interpretability. The selected components collectively represent the most informative aspects of the data, facilitating a more efficient and meaningful analysis. Use the scale function to make all firms completely comparable. The target is to find a combination that contains at least 95% of the original set.

```
scaled_data <- scale(data_cleaned_without_time)
# view(scaled_data)
#PCA analysis
pca_result <- prcomp(scaled_data)

# Used cumulative proportion to select number of the primary components
cumulative_proportion <- cumsum(pca_result$sdev^2) / sum(pca_result$sdev^2)
num_components <- which(cumulative_proportion >= 0.95)[1]

# Get the first num of components
pca_components <- pca_result$x[, 1:num_components]
```

```r
# Draw the proportion
plot(cumulative_proportion, type = "b", xlab = "Number of Components", ylab = "Cumulative Proportion of
```




```r
# print the num_components
cat("Number of Principal Components to Explain 95% of Variance:", num_components, "\n")
```

```
## Number of Principal Components to Explain 95% of Variance: 7
```

```r
# Print the summary of the results and write it into excel file
summary(pca_result)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     7.5200 2.70642 2.32145 1.84699 1.50673 1.11601 0.96266
## Proportion of Variance 0.6982 0.09043 0.06653 0.04212 0.02803 0.01538 0.01144
## Cumulative Proportion  0.6982 0.78859 0.85512 0.89724 0.92526 0.94064 0.95208
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.85219 0.74845 0.59649 0.50022 0.48128 0.44378 0.41376
## Proportion of Variance 0.00897 0.00692 0.00439 0.00309 0.00286 0.00243 0.00211
## Cumulative Proportion  0.96105 0.96796 0.97235 0.97544 0.97830 0.98073 0.98285
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.37438 0.36744 0.32672 0.29738 0.27288 0.24747 0.24610
## Proportion of Variance 0.00173 0.00167 0.00132 0.00109 0.00092 0.00076 0.00075
```

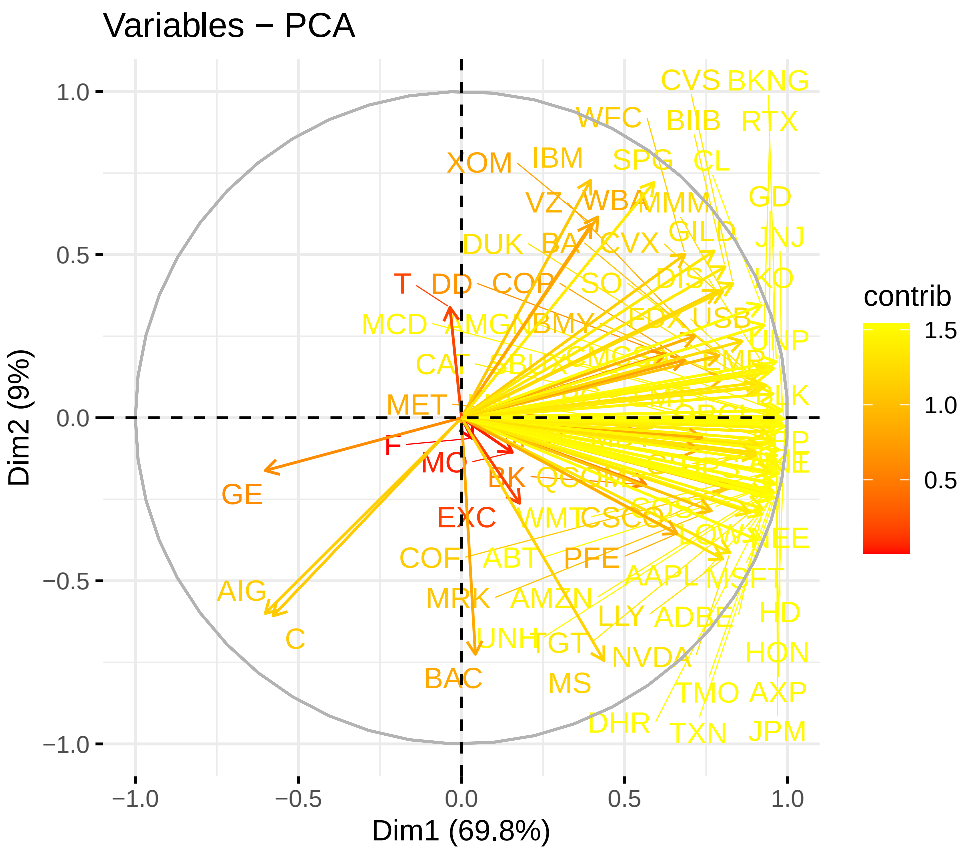
```
## Cumulative Proportion  0.98458 0.98625 0.98756 0.98866 0.98957 0.99033 0.99108
##                           PC22    PC23   PC24    PC25    PC26    PC27   PC28
## Standard deviation     0.23136 0.22539 0.2199 0.19287 0.18804 0.18135 0.1809
## Proportion of Variance 0.00066 0.00063 0.0006 0.00046 0.00044 0.00041 0.0004
## Cumulative Proportion  0.99174 0.99237 0.9930 0.99342 0.99386 0.99427 0.9947
##                           PC29    PC30    PC31    PC32   PC33    PC34    PC35
## Standard deviation     0.17740 0.16444 0.16255 0.15745 0.1546 0.14213 0.13776
## Proportion of Variance 0.00039 0.00033 0.00033 0.00031 0.0003 0.00025 0.00023
## Cumulative Proportion  0.99506 0.99539 0.99572 0.99602 0.9963 0.99657 0.99680
##                           PC36   PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.13432 0.1278 0.12216 0.11933 0.11558 0.11044 0.10515
## Proportion of Variance 0.00022 0.0002 0.00018 0.00018 0.00016 0.00015 0.00014
## Cumulative Proportion  0.99703 0.9972 0.99741 0.99759 0.99775 0.99790 0.99804
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.10272 0.10040 0.09601 0.09413 0.09273 0.08806 0.08509
## Proportion of Variance 0.00013 0.00012 0.00011 0.00011 0.00011 0.00010 0.00009
## Cumulative Proportion  0.99817 0.99829 0.99841 0.99852 0.99862 0.99872 0.99881
##                           PC50    PC51    PC52    PC53    PC54    PC55    PC56
## Standard deviation     0.08358 0.07899 0.07686 0.07492 0.07243 0.07124 0.06981
## Proportion of Variance 0.00009 0.00008 0.00007 0.00007 0.00006 0.00006 0.00006
## Cumulative Proportion  0.99889 0.99897 0.99904 0.99911 0.99918 0.99924 0.99930
##                           PC57    PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.06841 0.06387 0.06151 0.06121 0.05892 0.05678 0.05449
## Proportion of Variance 0.00006 0.00005 0.00005 0.00005 0.00004 0.00004 0.00004
## Cumulative Proportion  0.99936 0.99941 0.99946 0.99950 0.99955 0.99958 0.99962
##                           PC64    PC65    PC66    PC67    PC68    PC69    PC70
## Standard deviation     0.05275 0.05194 0.05179 0.04952 0.04774 0.04640 0.04412
## Proportion of Variance 0.00003 0.00003 0.00003 0.00003 0.00003 0.00003 0.00002
## Cumulative Proportion  0.99966 0.99969 0.99972 0.99975 0.99978 0.99981 0.99983
##                           PC71    PC72    PC73    PC74    PC75    PC76    PC77
## Standard deviation     0.04235 0.04073 0.03982 0.03894 0.03713 0.03616 0.03484
## Proportion of Variance 0.00002 0.00002 0.00002 0.00002 0.00002 0.00002 0.00001
## Cumulative Proportion  0.99985 0.99987 0.99989 0.99991 0.99993 0.99995 0.99996
##                           PC78    PC79    PC80    PC81
## Standard deviation     0.03227 0.02993 0.02656 0.02363
## Proportion of Variance 0.00001 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99997 0.99998 0.99999 1.00000
```

```r
pca_summary <- summary(pca_result)
```

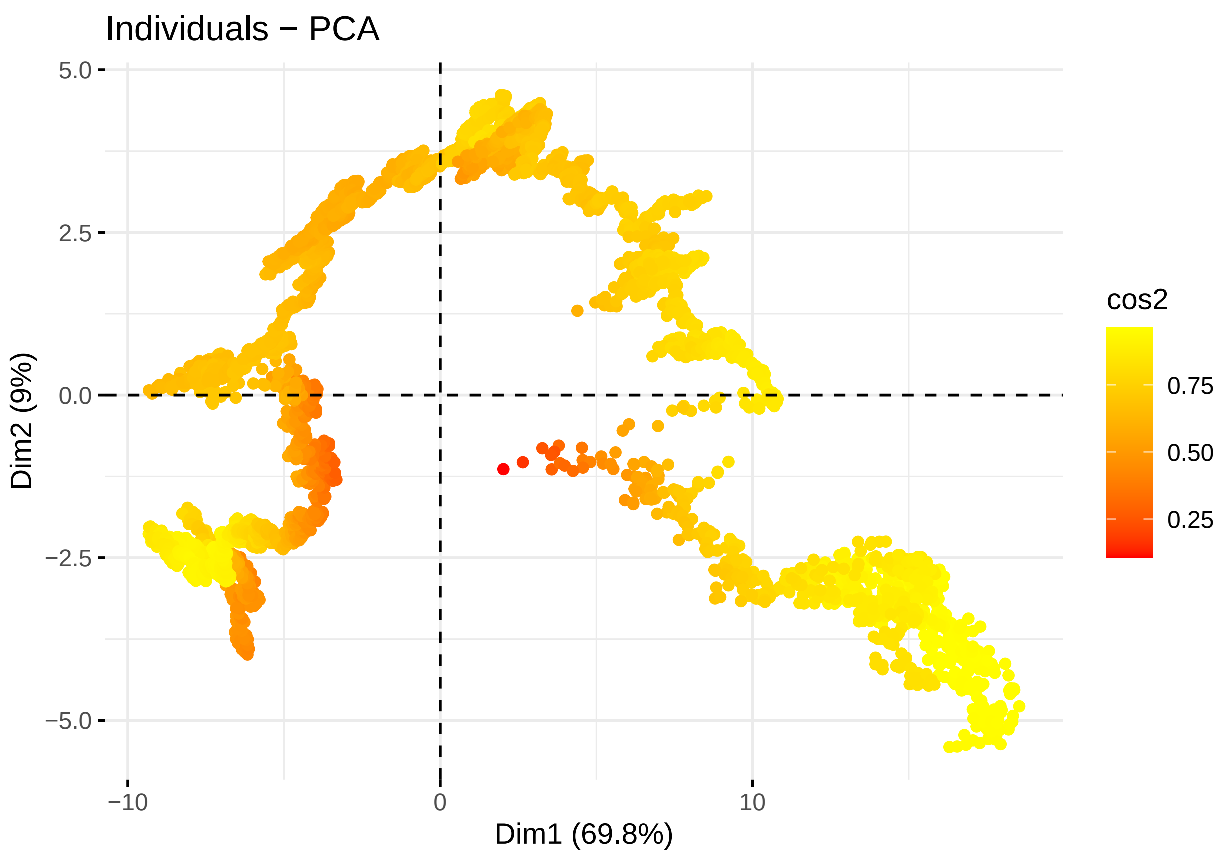
The number of principal components to explain 95% of variance is 7. We will choose the first 7 components as the principle components as these components explain about 95% of the variance. The plot of the cumulative proportion of variance explained by the principal components can be seen in the Figure.

The variance circle can visually summarize the contributions and relationships of original variables in the principal components. It provides a concise representation of each variable's contribution to the variance, with the distance from the circle center indicating the magnitude of the contribution. The color gradient on the circle signifies the level of contribution, aiding in the identification of variables with significant impact. The direction of variables on the circle reflects their correlation with the principal components. The distribution, relationships, and contributions of individual samples in PCA are presented in the individual PCA plot.

```r
# variance circle
fviz_pca_var(pca_result, col.var = "contrib",
                                gradient.cols = c("red", "yellow"),
                                repel = TRUE)
```




```r
#individual pca plot
fviz_pca_ind(pca_result, col.ind = "cos2",
                                    gradient.cols = c("red", "yellow"),
                                    repel = TRUE,
                                    geom.ind = "point", # Use points to represent individuals
                                    geom.var = "arrow", # Use arrows to represent variables
                                    axes = c(1, 2), # Display only the first two principal components
                                    ggtheme = theme_minimal(), # Set plot theme
                                    label = "var", # Label variables
                                    select.var = list(contrib = 2), # Select top contributors for label
)
```



PCA transforms the original data into seven columns of data in a new coordinate system, with roughly similar features(95.2% of the original variance.). The reduced-dimensional data is effective for following analysis, improving the speed of model training and the accuracy of predictions. However, during this process, some features of the original data will inevitably be lost. Therefore, in future work, for more effective utilization of data features, we can focus on more powerful feature extraction and feature learning models, such as convolutional neural networks, deep learning, and others.

3 Data Reduction

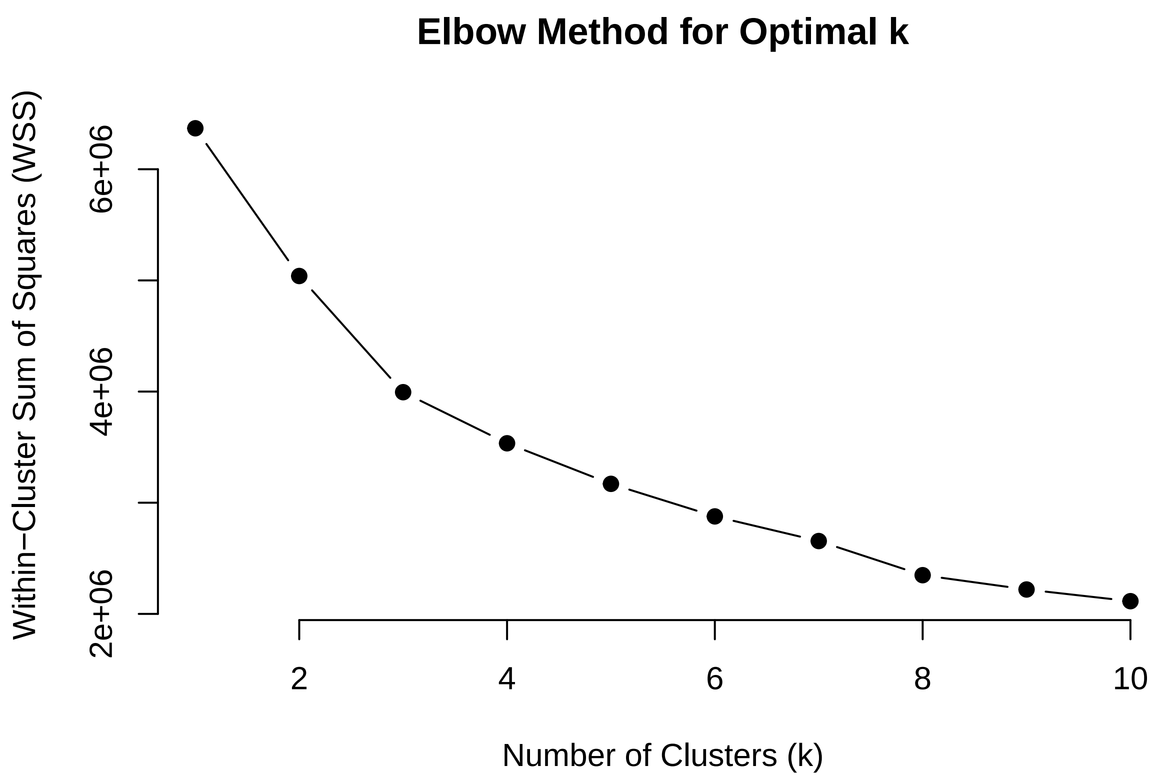
3.1 Clustering Date

When facing a very large data set (5786 observations), K-means becomes the only choice. First, use the total within-cluster sum square(WSS) to conduct the elbow test. The point with the last large drop of WSS would be the optimal k. To make different firms fully comparable, change the original dataset to proportional daily loss.

```
data_lossp=data_cleaned

diff=0
for (i in 2:length(data_cleaned)) {
    for (j in 2:length(data_cleaned[,1])) {

      data_lossp[j-1,i]=data_loss[j-1,i]/data_cleaned[j,i]

    }
```

```
  data_lossp[length(data_cleaned[,1]),i]=NA

}
data_lossp=na.omit(data_lossp)

#head(data_lossp)##The proportional loss data.
```

```
set.seed(20108)
wss=c()

for (i in 1:10) {
  kmeans_model <- kmeans(data_loss[, 2:82], centers = i)
  wss[i] <- kmeans_model$tot.withinss
}

plot( wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of Clusters (k)", ylab = "Within-Cluster Sum of Squares (WSS)",
     main = "Elbow Method for Optimal k")
```



The last large drop occurred on k=3. Then we cluster the date into three centers using kmeans. The code below also printed the number of days in each cluster, and the mean in every cluster.

```
set.seed(20108)
a2= kmeans(data_lossp[, 2:82], 3)
table(a2$cluster)
```

```
##
##    1    2    3
## 1026  763 3996
```

```
a2$centers
```

```
##            AAPL            VZ           WBA           WFC           WMT
## 1 -0.0193115425 -0.0100167938 -0.0117936996 -0.0216291033 -0.0096283953
## 2  0.0234959728  0.0132048356  0.0154592267  0.0271740057  0.0108450271
## 3 -0.0003590464  0.0002825918  0.0003195798  0.0005979772  0.0003043215
##             XOM           ABT          ADBE           AIG          AMGN
## 1 -1.327007e-02 -0.0098465903 -0.0212474927 -0.0227156845 -1.196060e-02
## 2  1.730974e-02  0.0129345164  0.0265195642  0.0367001391  1.634051e-02
## 3  5.915752e-05 -0.0002021905  0.0002065195  0.0006722851 -9.869454e-05
##             AMT          AMZN           AXP            BA           BAC
## 1 -1.534368e-02 -2.103230e-02 -0.0228524157 -1.802385e-02 -0.0249581846
## 2  2.211615e-02  2.603086e-02  0.0296968588  2.363991e-02  0.0331637053
## 3 -6.954161e-05  8.982491e-06  0.0002443945  7.468553e-05  0.0005873697
##            BIIB            BK          BKNG           BLK           BMY
## 1 -1.620738e-02 -0.0216750095 -0.0245190828 -1.949023e-02 -9.809453e-03
## 2  2.241777e-02  0.0284809219  0.0308433569  2.407726e-02  1.365476e-02
## 3 -3.629207e-05  0.0005225571  0.0008077618 -3.580514e-05  5.894597e-05
##               C           CAT            CL         CMCSA           COF
## 1 -0.0257409477 -0.0200298653 -7.509149e-03 -1.603881e-02 -0.0276909737
## 2  0.0366746829  0.0236175148  9.376688e-03  2.156877e-02  0.0349172942
## 3  0.0008513587  0.0002631815 -1.761336e-06 -3.162019e-05  0.0008894438
##             COP          COST          CSCO           CVS           CVX
## 1 -0.0151787007 -1.302910e-02 -0.0200760280 -1.135807e-02 -0.0134708628
## 2  0.0207861642  1.498056e-02  0.0250540164  1.497674e-02  0.0180920054
## 3 -0.0001739696  6.374117e-06  0.0008113827 -5.922882e-06 -0.0001014003
##              DD           DHR           DIS           DUK           EMR
## 1 -0.0201003526 -0.0150392089 -0.0177457716 -7.914655e-03 -0.0189748734
## 2  0.0259367122  0.0176352720  0.0227361510  1.070659e-02  0.0225392404
## 3  0.0004503516 -0.0001498928  0.0002521328  1.939236e-05  0.0005443501
##             EXC             F           FDX            GD           GE
## 1 -0.0087486227 -0.0206396387 -0.0184828842 -0.0130648176 -0.018088463
## 2  0.0125892605  0.0278437718  0.0211803622  0.0166566334  0.025580243
## 3 -0.0002127241  0.0006472432  0.0005303345 -0.0001735757  0.000339545
##            GILD            GS            HD           HON           IBM
## 1 -0.0152575346 -0.0228295621 -0.0166878558 -1.868053e-02 -0.0144097318
## 2  0.0186623225  0.0280610382  0.0208461225  2.355809e-02  0.0176825473
## 3 -0.0001084375  0.0005793706  0.0001377383  9.779547e-05  0.0004379242
##            INTC           JNJ           JPM            KO          LIN
## 1 -0.0200334123 -0.0074782737 -0.0246910309 -8.215203e-03 -0.016062427
## 2  0.0258355307  0.0100463549  0.0307129329  1.075054e-02  0.019041308
## 3  0.0007961094 -0.0001988896  0.0006149498 -1.199132e-05 -0.000047817
##             LLY           LMT           LOW           MCD           MDT
## 1 -1.026680e-02 -0.0096398378 -0.0179818817 -0.0096073003 -0.0113662508
## 2  1.275052e-02  0.0123349190  0.0208081025  0.0123126531  0.0144135123
## 3  2.423803e-05 -0.0004374783  0.0002046459 -0.0002996092  0.0002071668
##             MET          MMM            MO           MRK           MS
## 1 -0.0229806695 -0.014188030 -0.0075796168 -0.0098855648 -0.027971992
## 2  0.0292064137  0.017448812  0.0102056185  0.0128934667  0.035768255
## 3  0.0004903747  0.000264023  0.0003027071  0.0001557942  0.001002067
```

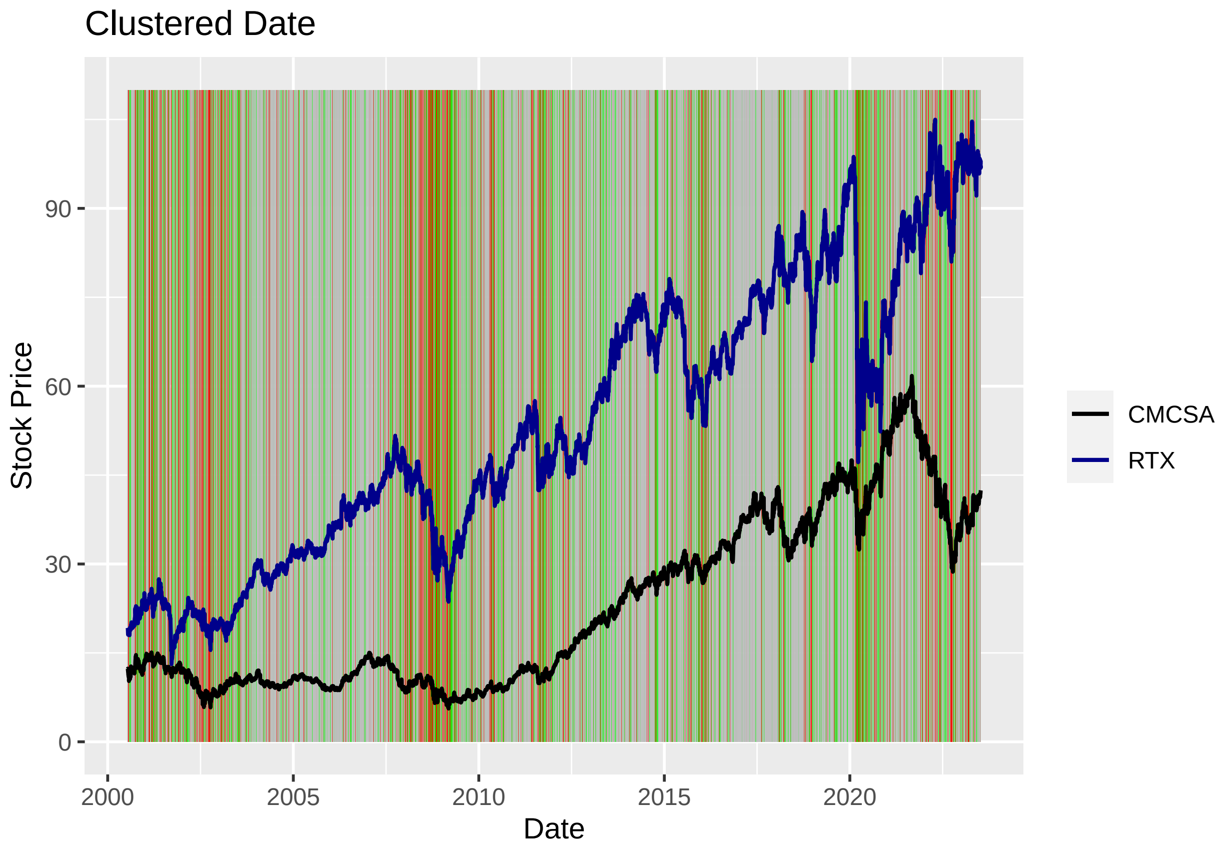
```
##             MSFT           NEE           NKE          NVDA          ORCL
## 1 -1.686964e-02 -0.0083595369 -0.0147777076 -0.0278224267 -0.0189222754
## 2  2.116315e-02  0.0109004458  0.0184155025  0.0371244372  0.0238220963
## 3  1.069095e-05 -0.0003765693 -0.0002148475 -0.0002438003  0.0004024117
##             PEP           PFE            PG          QCOM           RTX
## 1 -0.0075160800 -0.0104438984 -0.0072755126 -0.0200222202 -0.0164933371
## 2  0.0094867456  0.0140994685  0.0093240159  0.0252829877  0.0205656123
## 3 -0.0001535406  0.0002162825 -0.0002305004  0.0004352529  0.0001192981
##            SBUX            SO           SPG             T           TGT
## 1 -0.0170834390 -0.0063853150 -0.0180393585 -0.0107633681 -0.0152902462
## 2  0.0202258610  0.0081664058  0.0236656540  0.0150009463  0.0188352381
## 3  0.0000601603 -0.0001908436  0.0001092462  0.0002908502  0.0002678079
##             TMO           TXN           UNH           UNP           UPS
## 1 -0.0148704525 -0.0213214537 -0.0117645606 -1.619857e-02 -0.0132732119
## 2  0.0183057081  0.0259217421  0.0150097450  1.897759e-02  0.0151625098
## 3 -0.0002330211  0.0006837796 -0.0005108609  3.744534e-06  0.0003891343
##             USB
## 1 -0.0202809936
## 2  0.0257785353
## 3  0.0004947097
```

```r
for(i in 1:3)print(mean(a2$centers[i,]))
```

```
## [1] -0.01597334
## [1] 0.02044602
## [1] 0.0001758006
```

Recall the definition of loss, the first cluster indicates days that most stock prices are growing, the second cluster means dropping, and the last cluster means very small(or close to zero) dropping. Then draw the clustered date graph.

```r
background=rep(110,5785)
myColor=c("green","red","grey",rep(NA,5782))



ggplot(data_cleaned[2:5786,], aes(as.Date.character(data_cleaned[2:5786,1]))) +
  geom_bar(aes(y = background),fill=myColor[a2$cluster], stat = "identity", alpha = 0.7) +
  geom_line(aes(y = data_cleaned[2:5786,25],colour="CMCSA"),linewidth = 0.7) +
  geom_line(aes(y = data_cleaned[2:5786,71],colour="RTX"),linewidth = 0.7) +
  scale_colour_manual("",
                      breaks = c("CMCSA","RTX"),
                      values = c("black", "darkblue"))+
  xlab("Date") +
  scale_y_continuous("Stock Price", limits = c(0,110)) +
  labs(title="Clustered Date")
```



The red parts indicate days that most stock prices dropping(cluster 2), and green for growing(cluster 1). Grey for almost zero dropping. We found major red parts consisting of the US economic history: Dotcom Bubble recession period(2002-2003), the 2007–2008 financial crisis, the 2011 debt crisis, the 2015–2016 stock market selloff, the Covid-19 recession, and the 2023 United States Bank Failure. This means this clustering result successfully captured some major economic movements.

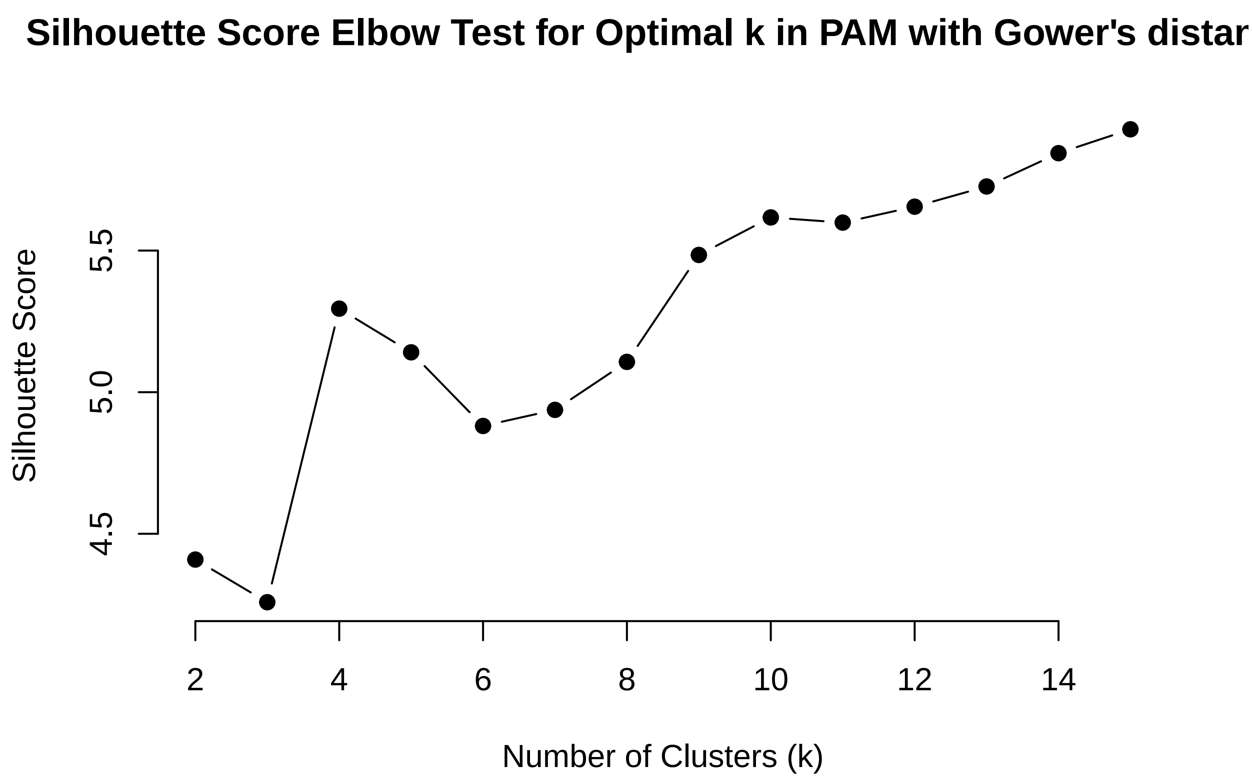
The two cluster centers in section 2.1 successfully showed growth in green parts and fall in red. It is good evidence that we can use the clustered data for analysis.

3.2 Diversify in Almost No Drop Days

Recall in 3.1, stock price may not move in the same direction during grey parts, these days should be more valuable for diversification. This part would conduct PAM with Gower's distance (similar to 2.1) again on grey days.

```
data_loss3=data_lossp[a2$cluster==3,]
data_loss3t=t((data_loss3[,2:82]))
dist=as.dist(gower.dist(data_loss3t))

silhouette_scores_pam <- function(k) {
  pam_model <- pam(dist, k = k,metric="manhattan")
  return(sum(silhouette(pam_model$clustering, dist)[,3]))
}

k_values <- 2:15  # Silhouette score is typically calculated for k >= 2
silhouette_values <- sapply(k_values, silhouette_scores_pam)
```

```r
plot(k_values, silhouette_values, type="b", pch = 19, frame=FALSE,
     xlab="Number of Clusters (k)", ylab="Silhouette Score",
     main="Silhouette Score Elbow Test for Optimal k in PAM with Gower's distance")
```



The graph found k=3 as the global minima. Therefore the optimal k should be 3.

```r
a=pam(dist,3,metric="manhattan")

table(a$clustering)
```

```
##
##  1  2  3
## 40 12 29
```

```r
a$medoids
```

```
## [1] "JNJ" "JPM" "MMM"
```

```r
data_loss3t=c()
```

This time there is no one firm cluster. The medoids of each cluster are JNJ, JPM, and MMM.

3.3 Diversify Suggestions

In terms of behavior, firms in the same clusters in 2.1 would more likely move together. Therefore may not be suitable for diverse risks. So diversity between clusters should be a better choice. PCA provides insight into where the variance comes from. So avoiding the most volatile firms would be more likely to secure the portfolio.

Investors have to be sensitive to the macroeconomic environment. The time to invest also plays a crucial role. The date cluster showed that most stock will drop in red parts, which means these risks are not diversifiable. Investments starting in the red parts may always receive profit when they enter green parts. If the investor has a very strong understanding of the macroenvironment, then trying the diversified strategy in 3.2 could be an option.

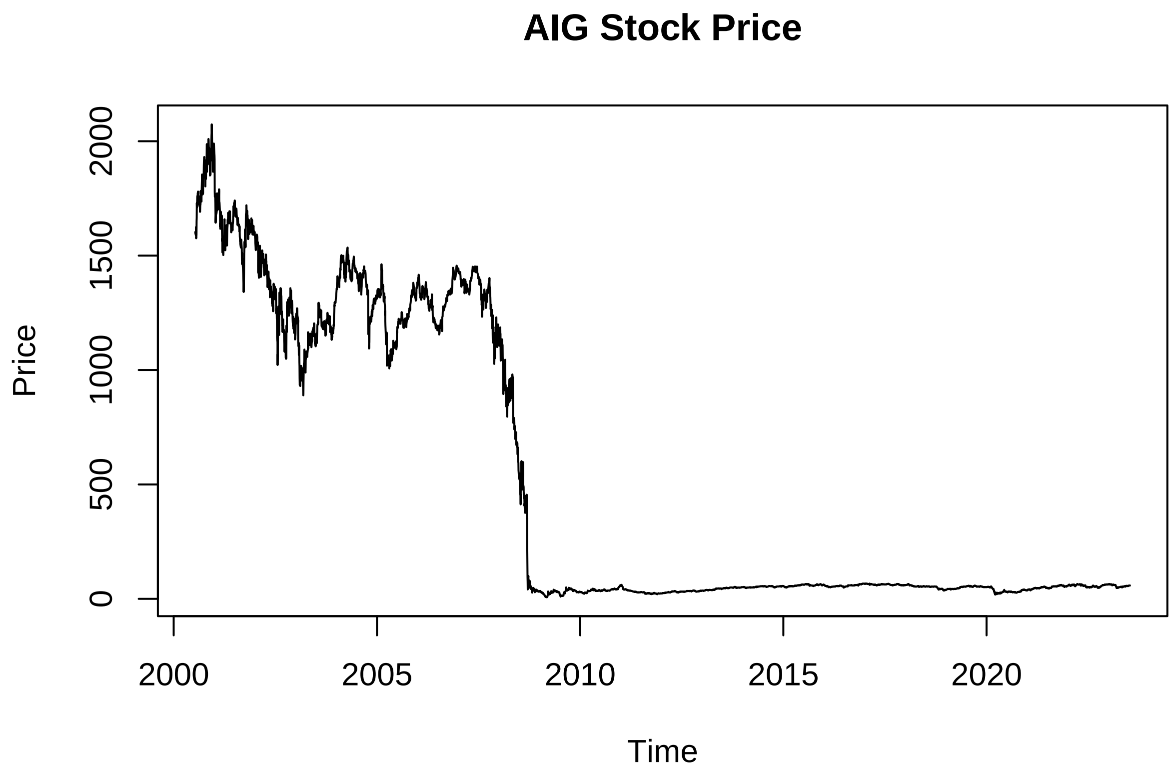
To predict the stock price, on the other hand, we need to pay attention to both volatile and representative firms. As the former is the main source of the variance, and the later provides the means of the stock price.

4 Time Series Modeling

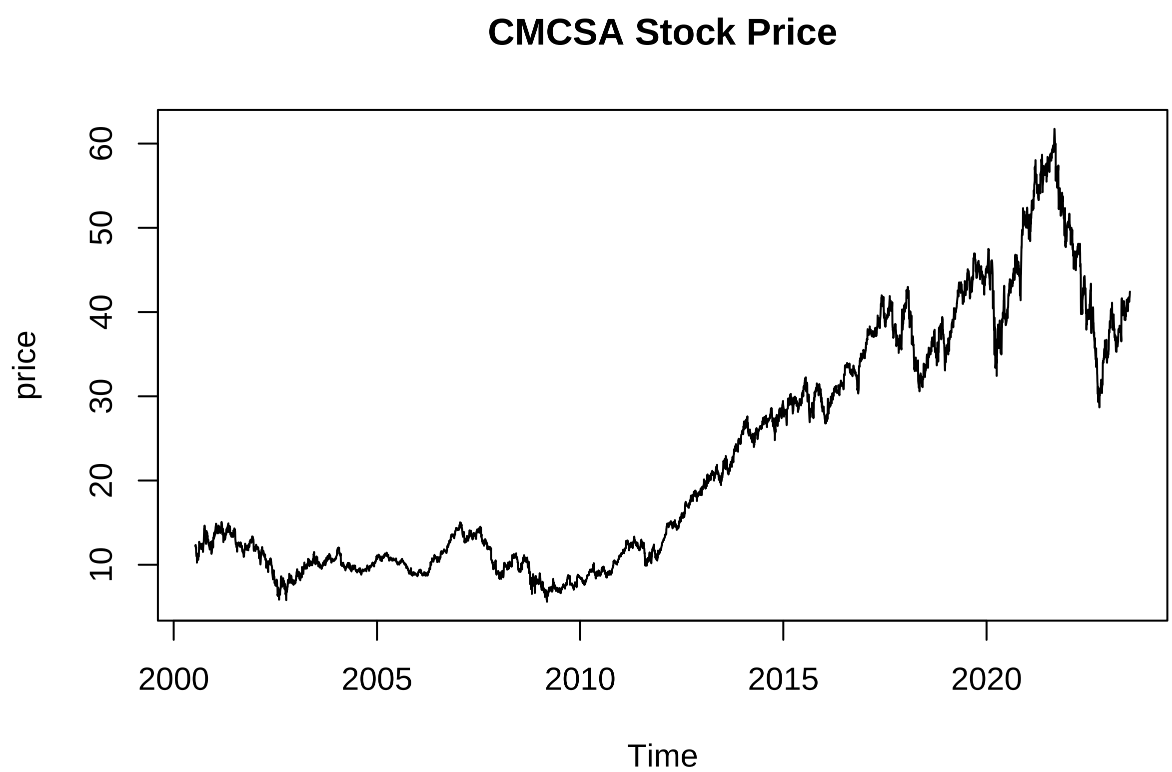
4.1 Visualization

This section will analyze CMCSA and AIG. First use visualization tools to demonstrate the nature of the data. And then try to apply several smoothing methods to the dataset. The purpose of doing smoothing is not only for data reduction, this method is also suitable for time series data analysis. As stock prices can be viewed as time series, we will implement smoothing on stock price and see what can be found.

```
AIG <- data.frame(time = 1:5786,as.Date.character(data_time),price = data_cleaned_without_time[,names(da
names(AIG)[2]="date"
plot(AIG[,2],AIG$price,type='l',main = "AIG Stock Price",ylab = "Price",xlab = "Time")
```

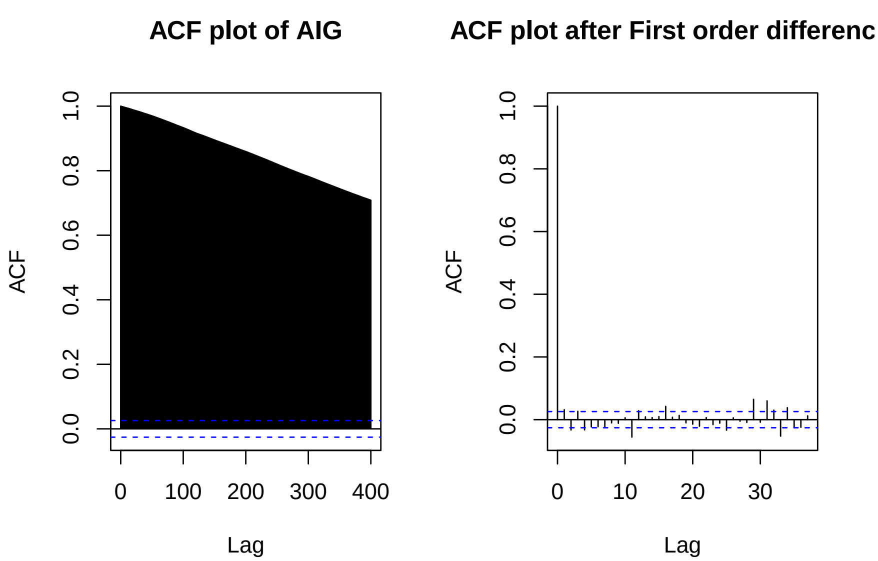



```
CMCSA= data.frame(time = 1:5786,as.Date.character(data_time),price = data_cleaned_without_time[,names(da
names(CMCSA)[2]="date"
plot(as.Date.character(CMCSA[,2]),CMCSA$price,type='l',main = "CMCSA Stock Price",ylab = "price",xlab =
```

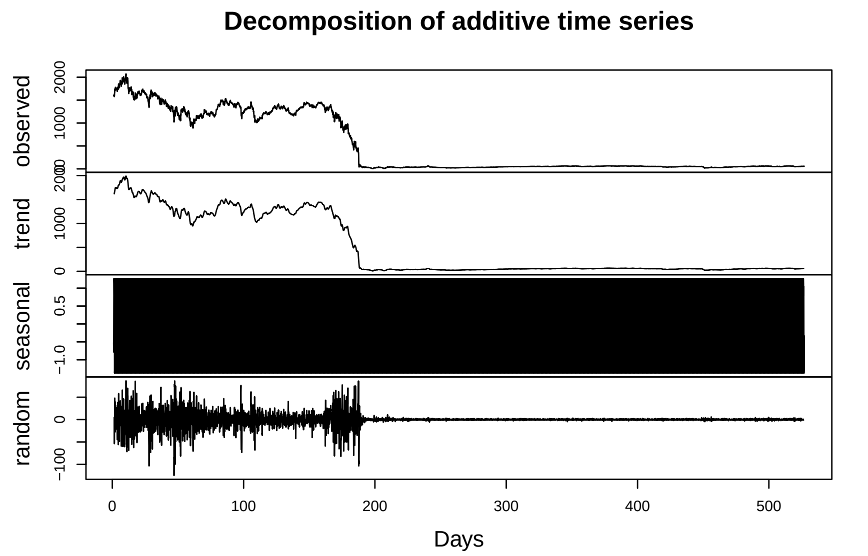


The two stocks showed a completely different shape and range. AIG ranges up to around 2000 while CMCSA ranges only up to 60. Now we try the decomposition and ACF plot for both stocks.

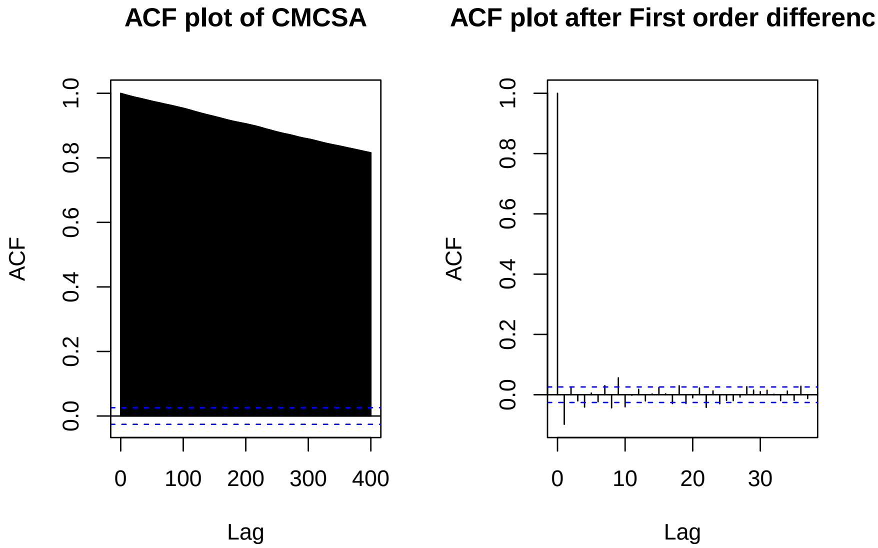
```
par(mfrow=c(1,2))
acf(AIG$price, lag.max = 400,main="ACF plot of AIG") ## strong trend, difference data
#pacf(AIG$price,main="PACF plot of AIG")
acf(diff(AIG$price),main="ACF plot after First order differencing")
```



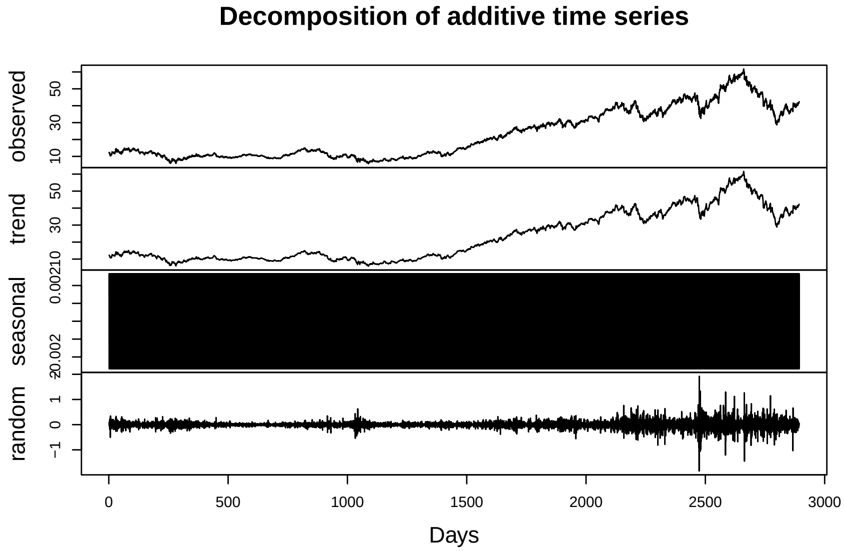
```r
plot(decompose(ts((AIG$price), start = 1, frequency = 11)),xlab = "Days")
```




```r
par(mfrow=c(1,2))
acf(CMCSA$price, lag.max = 400,main="ACF plot of CMCSA") ## strong trend, difference data
#pacf(CMCSA$price,main="PACF plot of CMCSA") ## strong trend, difference data
acf(diff(CMCSA$price),main="ACF plot after First order differencing")
```




```r
plot(decompose(ts((CMCSA$price), start = 1, frequency = 2)),xlab = "Days")
```

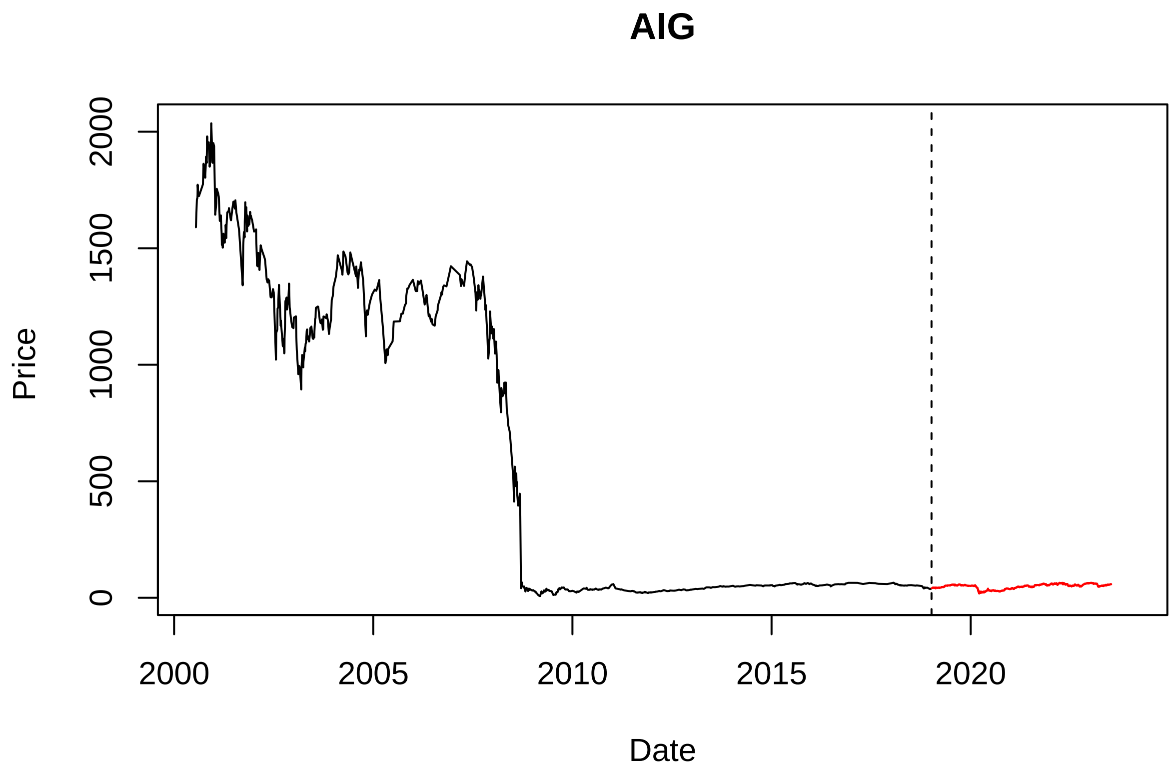


ACF plot shows there is a strong autocorrelation for both datasets. However, after we difference the dataset, we see that the correlation almost eliminated. AIG has positive autocorrelation on the 11th and CMCSA has a negative one on the second. Therefore, in the future fitting process, we may consider the first order difference, 11 for AIG and 2 for CMCSA as frequency. If we look at the decomposition graph, we first observe that the range of trend is almost the same as the original data. Then the range of seasonality is too narrow to make an inference. Therefore the trend should be an important part of analysis, and seasonality may not be significant. In the future, we may not consider the seasonal part.

4.2 Regression

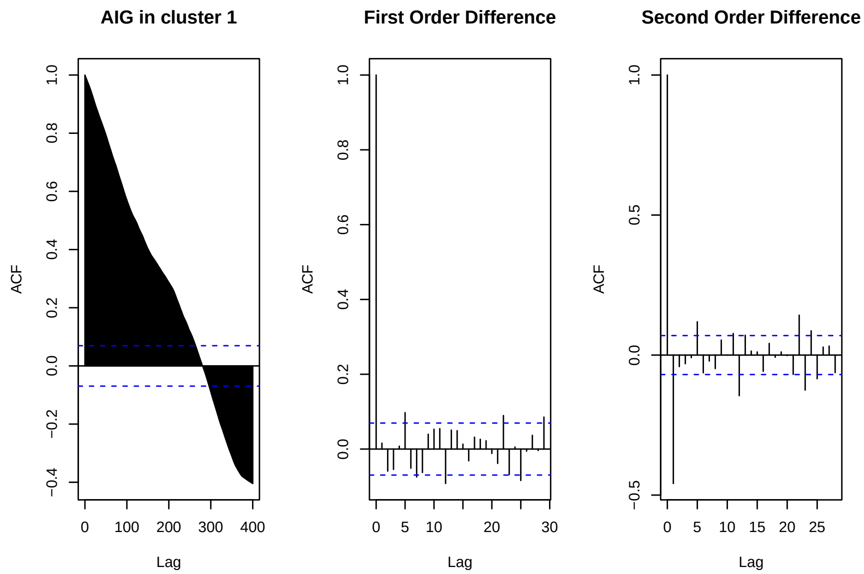
Now we can divide this dataset into two parts since this dataset is obtained from a long time period. One part is for training and another part is for testing. The training set contains the first 80% of observations. According to the conclusion in 4.1, the regression should based on the first-ordered difference. Here we again use the loss dataset(view it as a negative first-order difference).

```
AIG$loss=c(NA,data_loss[,names(data_loss)=="AIG"])
AIG$lossp=c(NA,data_lossp[,names(data_lossp)=="AIG"])
CMCSA$loss=c(NA,data_loss[,names(data_loss)=="CMCSA"])
CMCSA$lossp=c(NA,data_lossp[,names(data_lossp)=="CMCSA"])

first_part_AIG = AIG[1:4650,]
second_part_AIG = AIG[-(1:4650),]
first_part_CMCSA = CMCSA[1:4650,]
second_part_CMCSA = CMCSA[-(1:4650),]
first_part_AIG1 = first_part_AIG[a2$cluster[1:4650]==1,]
first_part_AIG2 = first_part_AIG[a2$cluster[1:4650]==2,]
first_part_AIG3 = first_part_AIG[a2$cluster[1:4650]==3,]
first_part_CMCSA1 = first_part_CMCSA[a2$cluster[1:4650]==1,]
first_part_CMCSA2 = first_part_CMCSA[a2$cluster[1:4650]==2,]
first_part_CMCSA3 = first_part_CMCSA[a2$cluster[1:4650]==3,]
plot(first_part_AIG1$date,first_part_AIG1$price,type='l', xlab = "Date", ylab = "Price ",
     xlim = c(11151, 19723),main = "AIG")
lines(second_part_AIG$date,second_part_AIG$price,col = "red")
abline(v = AIG$date[4650], lty = 2)
```

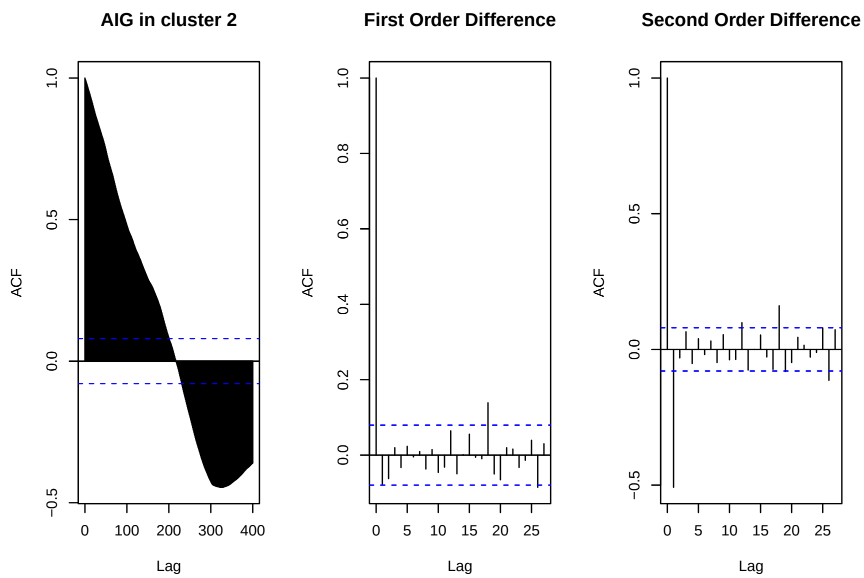


For this dataset, consider the first 4650 days as a training set and the rest as testing set, testing set is colored in red.

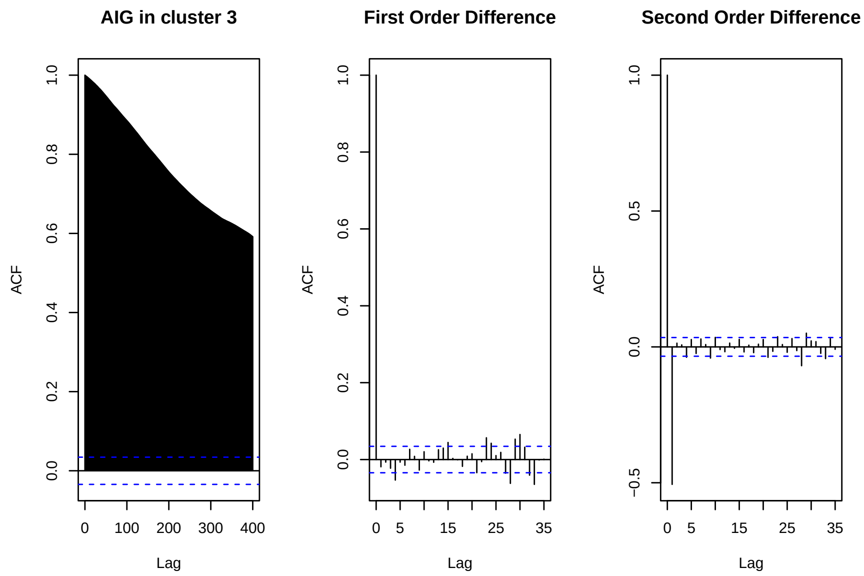
```
par(mfrow=c(1,3))
acf(first_part_AIG1$price, lag.max = 400, main="AIG in cluster 1")
acf(diff(first_part_AIG1$price), main="First Order Difference")
acf(diff(diff(first_part_AIG1$price)), main="Second Order Difference")
```



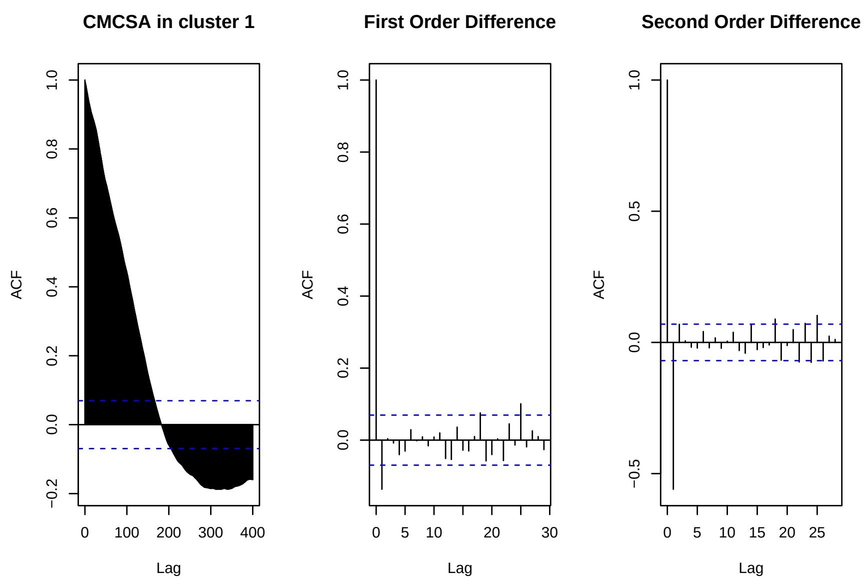

```
acf(first_part_AIG2$price, lag.max = 400, main="AIG in cluster 2")
acf(diff(first_part_AIG2$price), main="First Order Difference")
acf(diff(diff(first_part_AIG2$price)), main="Second Order Difference")
```



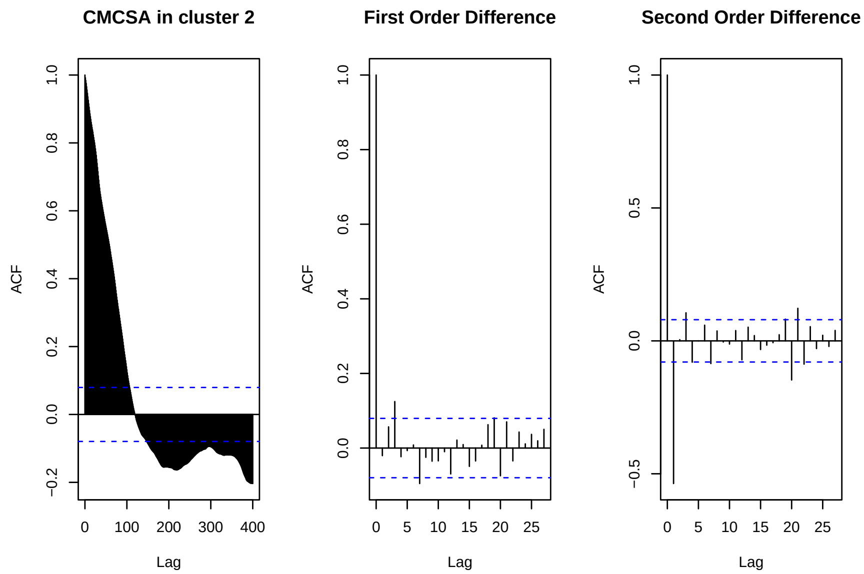

```
acf(first_part_AIG3$price, lag.max = 400, main="AIG in cluster 3")
acf(diff(first_part_AIG3$price), main="First Order Difference")
acf(diff(diff(first_part_AIG3$price)), main="Second Order Difference")
```



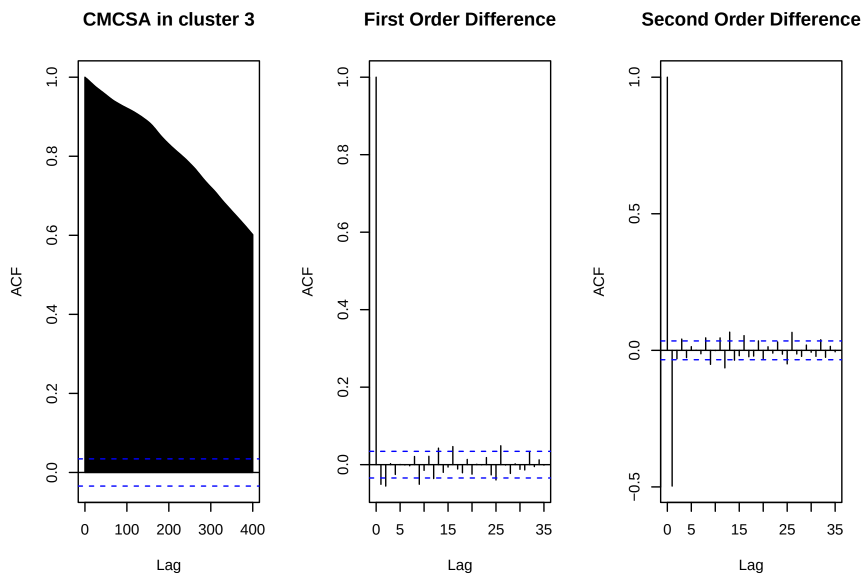

```r
acf(first_part_CMCSA1$price, lag.max = 400, main="CMCSA in cluster 1")
acf(diff(first_part_CMCSA1$price), main="First Order Difference")
acf(diff(diff(first_part_CMCSA1$price)), main="Second Order Difference")
```




```r
acf(first_part_CMCSA2$price, lag.max = 400, main="CMCSA in cluster 2")
acf(diff(first_part_CMCSA2$price), main="First Order Difference")
acf(diff(diff(first_part_CMCSA2$price)), main="Second Order Difference")
```




```r
acf(first_part_CMCSA3$price, lag.max = 400, main="CMCSA in cluster 3")
acf(diff(first_part_CMCSA3$price), main="First Order Difference")
acf(diff(diff(first_part_CMCSA3$price)), main="Second Order Difference")
```



Now, we do the same thing as we did previously, but only consider the training set. The cluster is based on the section 3.1. For AIG, cluster 1 part has the largest autocorrelation on the 5th lag, cluster 2 part on the 18th, and cluster 3 part on the 4th. CMCSA has autocorrelation on the 2nd, 4th, and 2nd lag in clusters 1, 2, and 3 respectively.

```r
dayset=function(periodlength=1){
  period_test=c()
  period_train=c()
  n=ceiling(5786/periodlength)
  full=rep(1:periodlength,n)
```

```r
  period_train<<-as.factor(full[1:4650])
  period_test<<-as.factor(full[4651:5786])
}
regtest=function(cluster){
  mse.reg = rep(0,10)
  mse.with.season = rep(0,10)
  index=(a2$cluster[1:4650]==cluster)
  day_train=day_train[index]
  period_train=period_train[index]
  for(i in 1:10){
  mod.reg1 = lm(train_data ~ poly(day_train, i))
  pred.reg1 = predict(mod.reg1, data.frame(day_train = day_test))
  mse.reg[i] = mean((pred.reg1 - test_data)**2)
  mod.reg2 = lm(train_data ~ poly(day_train, i) +period_train)
  pred.reg2 = predict(mod.reg2, data.frame(day_train = day_test,
  period_train = period_test))
  mse.with.season[i] = mean((pred.reg2 - test_data)**2)
  }
print((cbind(mse.reg, mse.with.season)) )
}
day_train = 1:4650
day_test = 4651:5786


test_data=second_part_AIG$lossp
train_data = first_part_AIG3$lossp
dayset(11)
regtest(3)
```

```
##            mse.reg mse.with.season
##  [1,] 0.0007395757    0.0007465626
##  [2,] 0.0007475021    0.0007540959
##  [3,] 0.0007478416    0.0007545521
##  [4,] 0.0010160834    0.0010262751
##  [5,] 0.0008169573    0.0008583815
##  [6,] 0.0041301209    0.0040480355
##  [7,] 0.0098426610    0.0092956826
##  [8,] 0.0126845011    0.0151049493
##  [9,] 0.1880645832    0.1362477891
## [10,] 1.9682405383    2.0426643147
```

```r
test_data=second_part_AIG$lossp
train_data = first_part_AIG2$lossp
dayset(18)
regtest(2)
```

```
##            mse.reg mse.with.season
##  [1,] 7.547709e-04    8.048978e-04
##  [2,] 7.417607e-04    7.935747e-04
##  [3,] 9.053080e-04    9.301913e-04
##  [4,] 9.084214e-04    8.026891e-04
##  [5,] 7.550792e-04    9.047319e-04
```

```
##   [6,] 3.958953e-02    4.222399e-02
##   [7,] 1.095703e-02    1.280492e-02
##   [8,] 3.909024e-01    3.326085e-02
##   [9,] 4.245380e+00    1.803744e+00
##  [10,] 7.031567e+01    4.609915e+01
```

```
test_data=second_part_AIG$lossp
train_data = first_part_AIG1$lossp
dayset(4)
regtest(1)
```

```
##             mse.reg mse.with.season
##   [1,] 8.024152e-04    8.248407e-04
##   [2,] 9.034280e-04    9.200109e-04
##   [3,] 9.045359e-04    1.016604e-03
##   [4,] 1.390366e-02    1.544338e-02
##   [5,] 1.141856e-03    2.203958e-03
##   [6,] 1.750147e-01    1.682220e-01
##   [7,] 3.299710e-01    2.851179e-01
##   [8,] 5.283266e+00    5.454603e+00
##   [9,] 1.963753e+01    1.442027e+01
##  [10,] 7.790670e+01    6.005790e+01
```

This is the mean square error for degrees up to 10 polynomial regressions for AIG. There are two categories; the First category on the left is without seasonality and the second one is with seasonality. We can find that the least me is when the degree equals one for cluster3, two for cluster2, and one for cluster1. All of those don't have seasonality.

```
test_data=second_part_CMCSA$lossp
train_data = first_part_CMCSA3$lossp
dayset(2)
regtest(3)
```

```
##             mse.reg mse.with.season
##   [1,] 0.0003206585    0.0003205931
##   [2,] 0.0003210717    0.0003210086
##   [3,] 0.0003214946    0.0003215782
##   [4,] 0.0003261821    0.0003262985
##   [5,] 0.0007168228    0.0006978836
##   [6,] 0.0004104126    0.0004246015
##   [7,] 0.0003311638    0.0003322228
##   [8,] 0.0397838527    0.0396552502
##   [9,] 0.2195142991    0.2348265749
##  [10,] 0.8382653475    0.8056975024
```

```
test_data=second_part_CMCSA$lossp
train_data = first_part_CMCSA2$lossp
dayset(4)
regtest(2)
```

```
##             mse.reg mse.with.season
##   [1,] 0.0003204929    0.0003247751
```

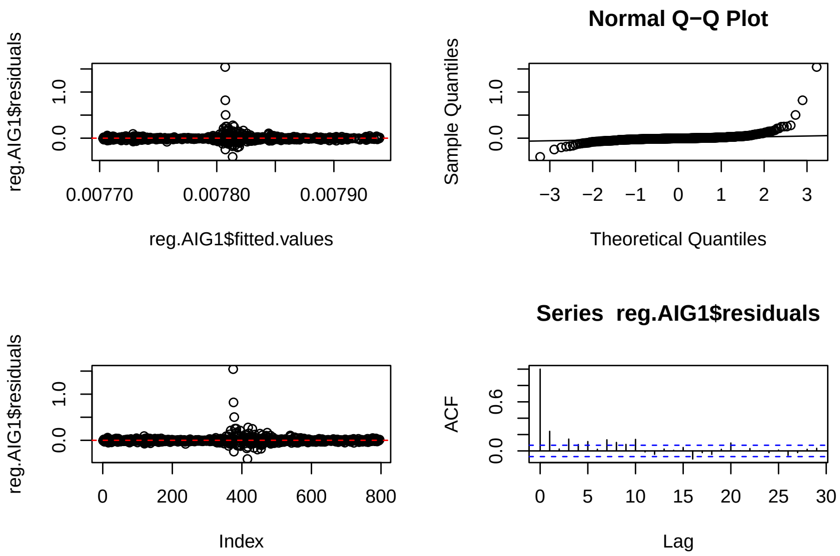
```
##   [2,] 0.0003707340    0.0003761250
##   [3,] 0.0003462507    0.0003545997
##   [4,] 0.0003489733    0.0003574817
##   [5,] 0.0010474068    0.0013076376
##   [6,] 0.0230363581    0.0200884471
##   [7,] 0.0139322136    0.0175887163
##   [8,] 0.0050557094    0.0019365967
##   [9,] 0.0037990639    0.0075576203
## [10,] 2.4465535056    1.6663656110
```

```
test_data=second_part_CMCSA$lossp
train_data = first_part_CMCSA1$lossp
dayset(2)
regtest(1)
```

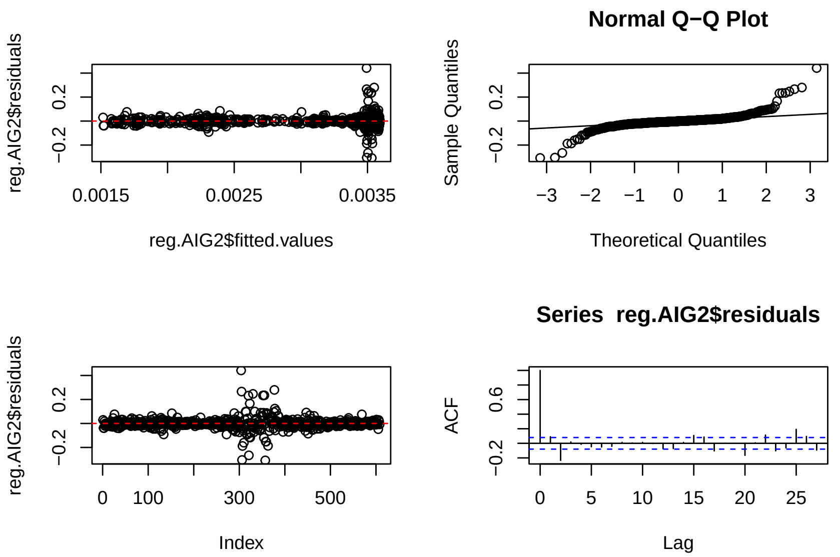
```
##            mse.reg mse.with.season
##   [1,] 3.206565e-04    3.214234e-04
##   [2,] 3.208354e-04    3.214933e-04
##   [3,] 5.421159e-04    5.588907e-04
##   [4,] 3.923771e-04    4.048879e-04
##   [5,] 8.812388e-04    1.023306e-03
##   [6,] 1.182872e-02    1.125811e-02
##   [7,] 3.673541e-04    4.055875e-04
##   [8,] 5.606160e-01    5.577544e-01
##   [9,] 1.638701e+01    1.590225e+01
## [10,] 8.638328e+00    8.113756e+00
```

The result for CMCSA is slightly different. In cluster 3, the least MSE occurs in order 1 with seasonality. The other two clusters all prefer to order one without seasonality. Once the model is selected, the residual diagnosis would be able to evaluate the performance of these models.

```
day1=day_train[(a2$cluster[1:4650]==1)]
day2=day_train[(a2$cluster[1:4650]==2)]
day3=day_train[(a2$cluster[1:4650]==3)]
reg.AIG1=lm(first_part_AIG1$lossp ~ poly(day1, 1))
reg.AIG2=lm(first_part_AIG2$lossp ~ poly(day2, 2))
reg.AIG3=lm(first_part_AIG3$lossp ~ poly(day3, 1))
dayset(2)
season=period_train[(a2$cluster[1:4650]==3)]
par(mfrow=c(2,2))
plot(reg.AIG1$fitted.values,reg.AIG1$residuals)
abline(h=0,lty=2,col="red")
qqnorm(reg.AIG1$residuals)
qqline(reg.AIG1$residuals)
plot(reg.AIG1$residuals)
abline(h=0,lty=2,col="red")
acf(reg.AIG1$residuals)
```

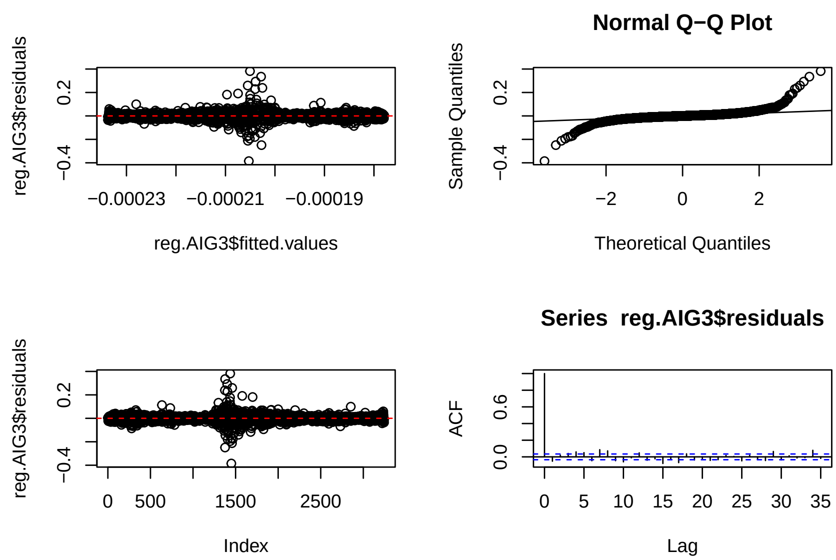



```r
par(mfrow=c(2,2))
plot(reg.AIG2$fitted.values,reg.AIG2$residuals)
abline(h=0,lty=2,col="red")
qqnorm(reg.AIG2$residuals)
qqline(reg.AIG2$residuals)
plot(reg.AIG2$residuals)
abline(h=0,lty=2,col="red")
acf(reg.AIG2$residuals)
```



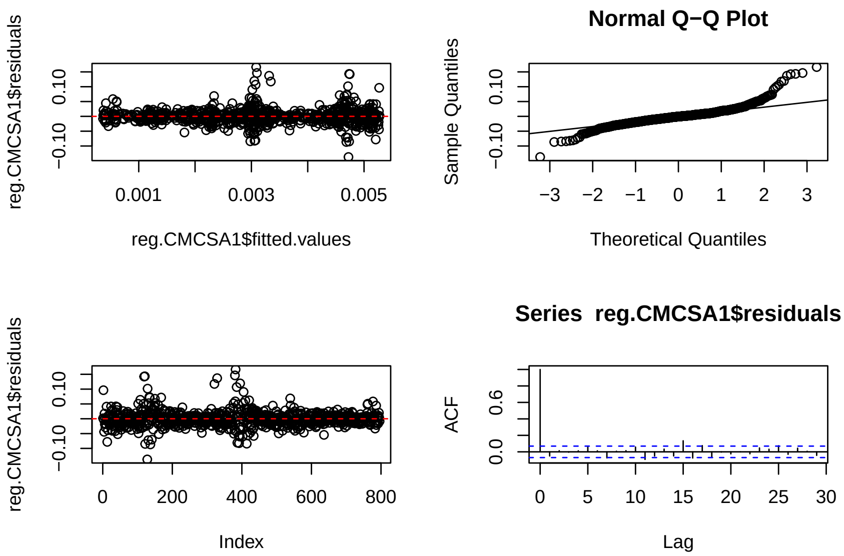

```r
par(mfrow=c(2,2))
plot(reg.AIG3$fitted.values,reg.AIG3$residuals)
abline(h=0,lty=2,col="red")
qqnorm(reg.AIG3$residuals)
```

```
qqline(reg.AIG3$residuals)
plot(reg.AIG3$residuals)
abline(h=0,lty=2,col="red")
acf(reg.AIG3$residuals)
```

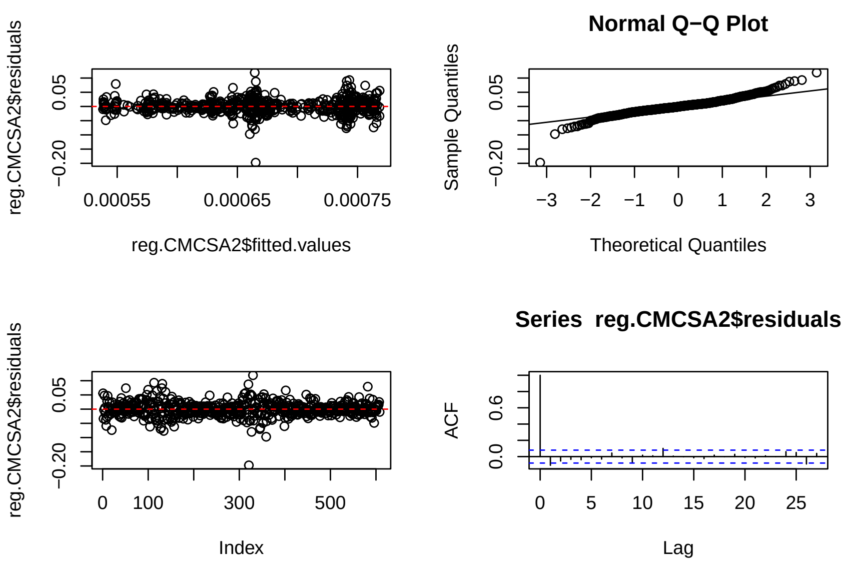


Three clusters produced similar results for AIG. ACF plot found that the residual has few trends remaining. Residual is evenly distributed around the zero line withot pattern, but the extreme occurs next to each other. The even distribution confirmed an accurate point estimation. The QQ plot showed the middle part is following normal, but the tail is much heavier than the normal distribution. All of those showed that the residual is not a normal white noise. Therefore if still assume the residual follows the normal distribution, the frequency and severity of extreme values would be significantly under estimated. The variance estimate may not be accurate enough.

```
reg.CMCSA1=lm(first_part_CMCSA1$lossp ~ poly(day1, 1))
reg.CMCSA2=lm(first_part_CMCSA2$lossp ~ poly(day2, 1))
reg.CMCSA3=lm(first_part_CMCSA3$lossp ~ poly(day3, 3)+season)
par(mfrow=c(2,2))
plot(reg.CMCSA1$fitted.values,reg.CMCSA1$residuals)
abline(h=0,lty=2,col="red")
qqnorm(reg.CMCSA1$residuals)
qqline(reg.CMCSA1$residuals)
plot(reg.CMCSA1$residuals)
abline(h=0,lty=2,col="red")
acf(reg.CMCSA1$residuals)
```

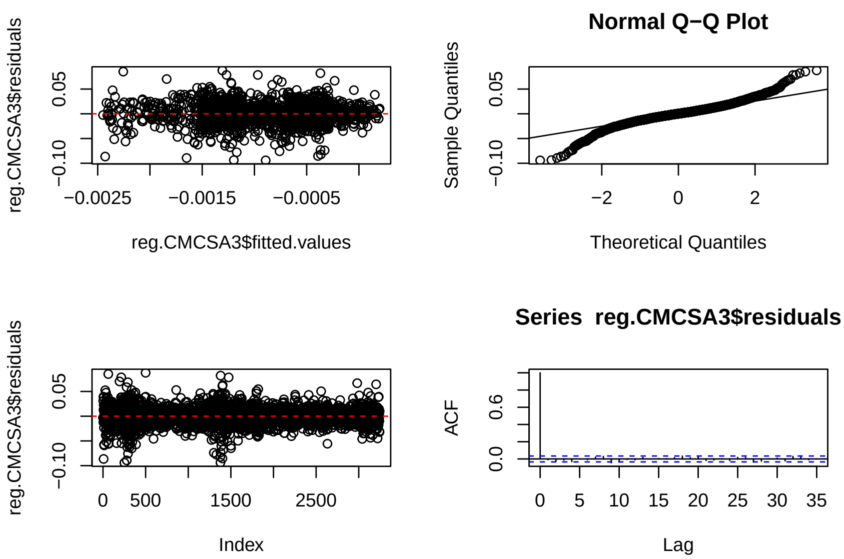



```r
par(mfrow=c(2,2))
plot(reg.CMCSA2$fitted.values,reg.CMCSA2$residuals)
abline(h=0,lty=2,col="red")
qqnorm(reg.CMCSA2$residuals)
qqline(reg.CMCSA2$residuals)
plot(reg.CMCSA2$residuals)
abline(h=0,lty=2,col="red")
acf(reg.CMCSA2$residuals)
```




```r
par(mfrow=c(2,2))
plot(reg.CMCSA3$fitted.values,reg.CMCSA3$residuals)
abline(h=0,lty=2,col="red")
qqnorm(reg.CMCSA3$residuals)
```

```r
qqline(reg.CMCSA3$residuals)
plot(reg.CMCSA3$residuals)
abline(h=0,lty=2,col="red")
acf(reg.CMCSA3$residuals)
```

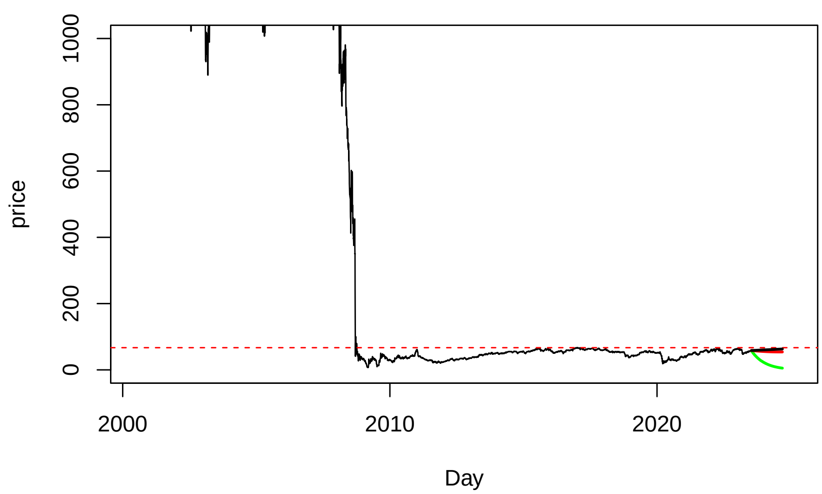


The regression model for CMCSA is similar to AIG. But the range of the residual reduced significantly. The QQ plot is also straighter, while tails are still heavier than normal distribution. CMCSA's ACF plots are very clean. So regression model works better on CMCSA than AIG. Overall the regression model successfully captures trends and some auto correlation, but it may not be suitable for variance estimation.

Then we use the model to make a prediction. The length of the prediction is 300 business days. There are three different prediction lines. The red line prediction assumes most stocks fall in following 300 business days(cluster 2 in section 3.1), the green line is most stocks grow(cluster 1), and the black line is most stocks almost no drop(cluster 3).

```r
time_new = 5786 +(1:300)
i=1;j=1
date_new=c()
date=""
while(i<=length(time_new)){
  date=as.Date(19551+j)
  if(isBizday(as.timeDate(date))){
    date_new=c(date_new,date)
    j=j+1
    i=i+1
  }
  else j=j+1
}
prediction1 = predict(reg.AIG1, data.frame(day1 = time_new))
prediction2 = predict(reg.AIG2, data.frame(day2 = time_new))
prediction3 = predict(reg.AIG3, data.frame(day3 = time_new))
i=1
price1=c();price2=c();price3=c()
while (i<=length(time_new)) {
```

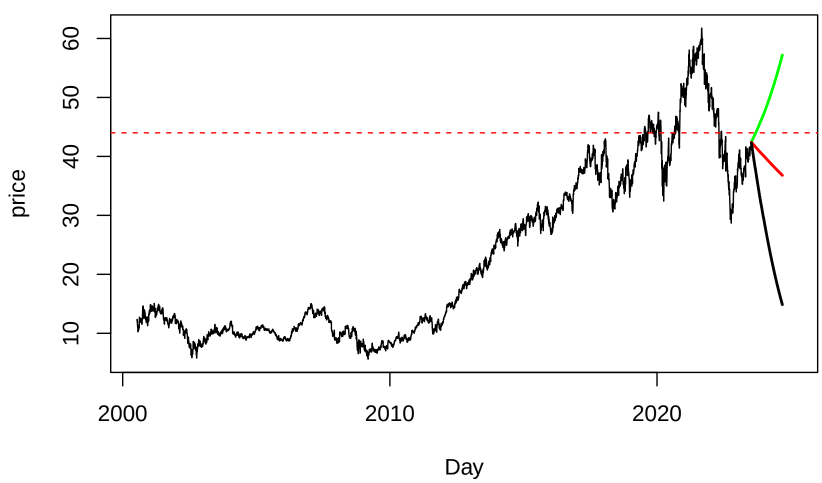
```
  if(i==1){
    price1[i]=58.45-prediction1[i]*58.45
    price2[i]=58.45-prediction2[i]*58.45
    price3[i]=58.45-prediction3[i]*58.45
  }
  else{
    price1[i]=price1[i-1]-prediction1[i]*price1[i-1]
    price2[i]=price2[i-1]-prediction2[i]*price2[i-1]
    price3[i]=price3[i-1]-prediction3[i]*price3[i-1]
  }
  i=i+1
}
plot(AIG$date,AIG$price, xlim = c(11151, 20100), type = "l",
    ylim = c(0,1000), xlab = "Day", ylab = "price")
abline(h=66.94,col = "red", lwd = 1,lty=2)
lines(as.Date(date_new), price1, col = "green", lwd = 2)
lines(as.Date(date_new), price2, col = "red", lwd = 2)
lines(as.Date(date_new), price3, col = "black", lwd = 2)
```



The black line is point estimation assuming the following 300 days close to almost zero drop(cluster 3 in section 3.1), red is almost all firm drop, and green is almost all firm grow. the red horizontal line is the real price on Dec. 23. The estimation of AIG shows it still has negative momentum due to historical fall. The current price ($66.94) was not included in the predictions. This may indicate this regression estimator is not the best choice for analyzing AIG. The negative macro environment poses fewer effects on it, which means it may potentially become a hedging vehicle.

```
prediction1 = predict(reg.CMCSA1, data.frame(day1 = time_new))
prediction2 = predict(reg.CMCSA2, data.frame(day2 = time_new))
  n=ceiling(300/2)
  full=rep(1:2,n)
  season<-as.factor(full[1:300])
prediction3 = predict(reg.CMCSA3, data.frame(day3 = time_new,season=season))
i=1
price1=c();price2=c();price3=c()
```

```
while (i<=length(time_new)) {
  if(i==1){
    price1[i]=42.425-prediction1[i]*58.45
    price2[i]=42.425-prediction2[i]*58.45
    price3[i]=42.425-prediction3[i]*58.45
  }
  else{
    price1[i]=price1[i-1]-prediction1[i]*price1[i-1]
    price2[i]=price2[i-1]-prediction2[i]*price2[i-1]
    price3[i]=price3[i-1]-prediction3[i]*price3[i-1]
  }
  i=i+1
}
plot(AIG$date,CMCSA$price, xlim = c(11151, 20100), type = "l",
    xlab = "Day", ylab = "price")
abline(h=44,col = "red", lwd = 1,lty=2)
lines(as.Date(date_new), price1, col = "green", lwd = 2)
lines(as.Date(date_new), price2, col = "red", lwd = 2)
lines(as.Date(date_new), price3, col = "black", lwd = 2)
```



CMCSA's estimation surprisingly has the black line to be the lowest line. The big difference between lines indicates less diversified value and the macro environment is critical when investing this stock. Again, confident interval from the regression model is not appropriate for modeling stocks, as normal assumption would significantly underestimate the tail distribution.

4.2 Smoothing

we want to transform the non-stationary data into stationary data using simple exponential smoothing, double exponential smoothing, additive smoothing, and multiplicative smoothing. Smoothing methods and the Holtwinter predictive method encode lots of values from the past and use them to predict "typical" values for the present and future. This section will use four different methods and select the one with the least MSE as the final smoothing method.

Mse of simple exponential smoothing

```
## [1] "AIG MSE"
```

```
## [1] 181.3418
```

```
## [1] "CMCSA MSE"
```

```
## [1] 130.7546
```

Mse of double exponential smoothing model

```
## [1] "AIG MSE"
```

```
## [1] 2460.153
```

```
## [1] "CMCSA MSE"
```

```
## [1] 117.7841
```

Mse of additive-HoltWinters model

```
## [1] "AIG MSE"
```

```
## [1] 4548.014
```

```
## [1] "CMCSA MSE"
```

```
## [1] 169.4187
```

Mse of multiplicative-HoltWinters model

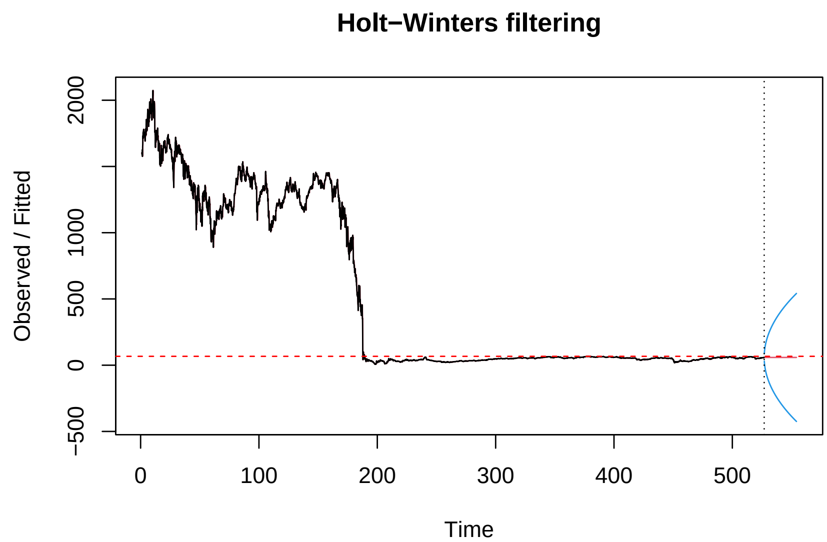
```
## [1] "AIG MSE"
```

```
## [1] 5036.477
```

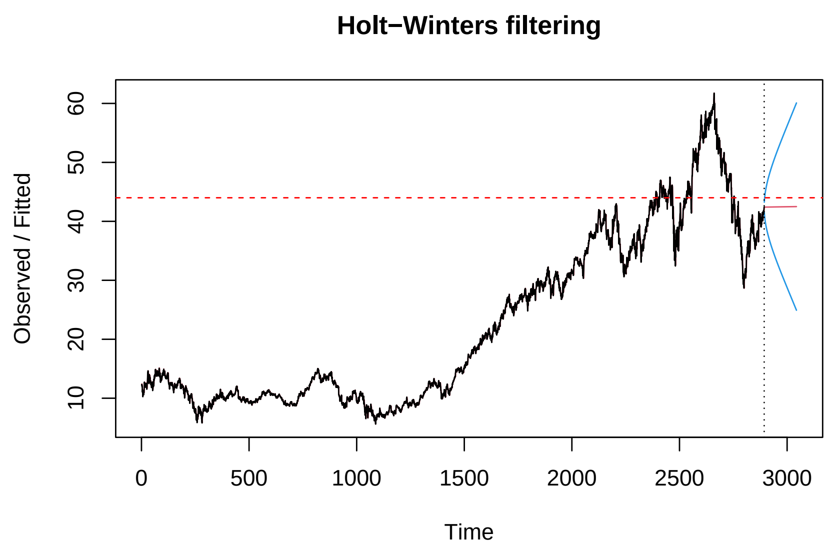
```
## [1] "CMCSA MSE"
```

```
## [1] 168.9677
```

```
ts1 = ts(AIG$price, start = 1, frequency = 11)
ts2 = ts(CMCSA$price, start = 1, frequency = 2)
final_smoothing <- HoltWinters(ts1, gamma = FALSE, beta = FALSE)
plot(final_smoothing, predict(final_smoothing, n.ahead = 300,
                              prediction.interval = TRUE))
abline(h=66.94,col = "red", lwd = 1,lty=2)
```




```
## due to amount of dataset, Holt wnter prediction overlap with original data.
##If we zoom in the plot, we can still observe red line.
final_smoothing <-  HoltWinters(ts2, gamma = FALSE)
plot(final_smoothing, predict(final_smoothing, n.ahead = 300,
                              prediction.interval = TRUE))
abline(h=44,col = "red", lwd = 1,lty=2)
```



The scale is large, if zoom in on the plot, we would be able to observe that the prediction interval is larger than the one produced by regression. It provides the potential growth range of AIG and successfully includes the current price. Knowing that stock price has fat-tail property, the Holt-winter prediction should be more reasonable.

4.3 Moldeling result

Knowing that the loss has heavy-tailed and autocorrelated properties, the normal distribution assumption in the regression estimate is not adequate. The scenario-based regression group instead produced prediction intervals for AIG and CMCSA. Regressions may capture the trend, scenario effect, and autocorrelations, but may oversimplify the extreme values and undiversifyable risks. Resulting in a narrower confidence interval for AIG and wider for CMCSA.

The time series method is also able to capture most autocorrelation and trends. The confidence interval is larger. AIG's confidence is wider than CMCSA's, which seems much more reasonable. However, Part of the confidence interval of AIG's stock is lower than zero, this may not happen in real life.

Note that scenario-based modeling and time series modeling are widely used, Despite the performance in this project is not ideal. Some more advanced models may improve the model significantly. Like extreme value theory, Generalized regression model, GARCH, and ARIMA time series.

5 Stock Price Prediction with Supervised Learning

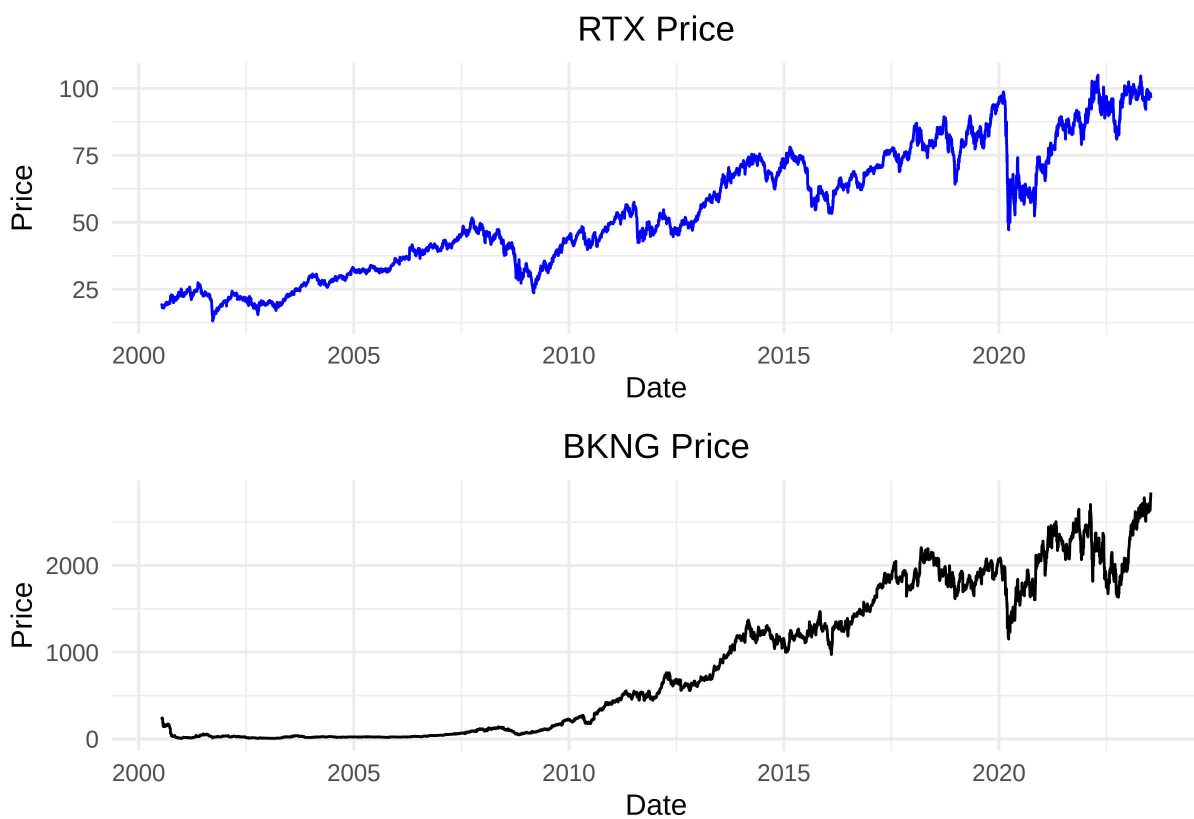
Besides clustering, forecasting stock prices can also be approached as a regression task. Here, regression techniques are utilized to discern and model the connections between various influential factors and the continuous variable of interest, namely the future stock prices. This method involves inputting historical information, such as previous stock prices and additional financial metrics, into a regression-based framework. The framework then examines this information to detect patterns, trends, and associations that aid in projecting forthcoming prices. The range of regression methods extends from basic linear regression, which presumes a straightforward linear link between the input variables and the stock price, to more sophisticated models like Support Vector Regression or neural networks. These advanced models can apprehend non-linear dynamics and intricate interdependencies within the data. Specifically, we will focus on constructing a recurrent neural network with LSTM (Long Short Term Memory) units. This approach is geared towards providing a foundational comprehension of predictions in time series analysis, particularly for univariate scenarios and non-stationary historical data.

5.1 Data analysis and preprocessing

As previously mentioned, RTX and BKNG have been chosen for modeling due to their position as the central element of one of the clusters, and the most volatile firm. Initially, we'll extract the closing stock prices from the broader dataset. The following graphs will illustrate the trajectory of the stock price for RTX and BKNG over time.

```r
df <- data_cleaned

# Select only the 'Date' and 'JNJ' columns and convert 'Date' to a date object
RTX_data <- df %>%
  select(Date, RTX)
RTX_data$Date <- as.character.Date(df$Date)# Set 'Date' as the index
RTX_data$Date <- as.Date(df$Date)
BKNG_data <- df %>%
  select(Date, BKNG)
BKNG_data$Date <- as.character.Date(df$Date)
BKNG_data$Date <- as.Date(df$Date)
#plot(RTX_data$Date,RTX_data$RTX,type='l',xlab="Date" ,ylab="Price" ,col="Blue",main="RTX price")
p1=ggplot(RTX_data, aes(x = Date, y = RTX)) +
  geom_line(color = "blue") +
```

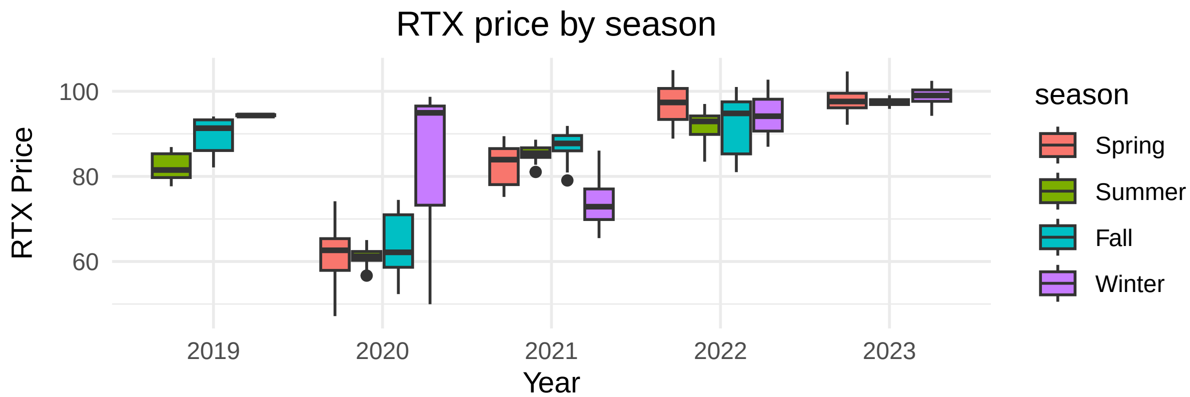
```
  ggtitle("RTX Price") +
  xlab("Date") + ylab("Price") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
p2=ggplot(BKNG_data, aes(x = Date, y = BKNG)) +
  geom_line(color = "black") +
  ggtitle("BKNG Price") +
  xlab("Date") + ylab("Price") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
library(patchwork)
p1+p2+plot_layout(ncol=1)
```

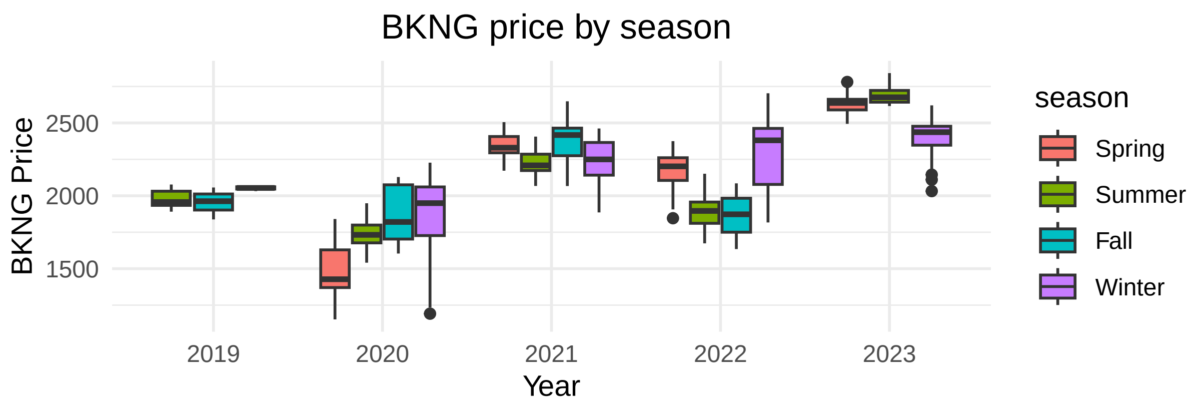


The variations in stock valuations are influenced by a wide array of elements, such as the operational success of a company, the moods and trends among investors, the overall state of the marketplace, and key economic signals. These valuations are constantly in change, shifting in response to the balance between supply and demand, and they are shaped by both the predictable and the unforeseen.

The following illustration is a box plot chart that aids in examining the range and shifts in stock valuations over time, segmented by seasons. This depiction spans from the years 2019 through 2023 and dissects the data into four timeframes: Spring, Summer, Fall, and Winter. Each rectangular section on the plot signifies the middle fifty percent of the price points for JNJ during each season, also known as the interquartile range (IQR), with a line inside demarcating the median price. The extensions, or 'whiskers', of these rectangles reach out to encompass most of the price data, setting aside the anomalies, which are marked as discrete dots.

```r
library(plotly)
date_processor <- function(df) {
  df <- df %>%
    mutate(date = as.Date(df$Date),
           dayofweek = wday(date, label = TRUE),
           weekday = wday(date, label = TRUE),
           month = month(date),
           year = year(date),
           dayofyear = yday(date),
           dayofmonth = mday(date),
           weekofyear = week(date),
           date_offset = (month(date)*100 + mday(date) - 320)%%1300,
           season = cut(date_offset, breaks = c(-1, 300, 602, 900, 1300),
                        labels = c('Spring', 'Summer', 'Fall', 'Winter')))
  return(df)
}

RTX_processed <- date_processor(RTX_data)
BKNG_processed <- date_processor(BKNG_data)

RTX_subset <- RTX_processed[4800:nrow(RTX_processed), ]
BKNG_subset <- BKNG_processed[4800:nrow(BKNG_processed), ]
p1 <- ggplot(RTX_subset, aes(x = as.factor(year), y = RTX, fill = season)) +
  geom_boxplot() +
  labs(title = " RTX price by season", x = "Year", y = "RTX Price") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title
p2 <- ggplot(BKNG_subset, aes(x = as.factor(year), y = BKNG, fill = season)) +
  geom_boxplot() +
  labs(title = " BKNG price by season", x = "Year", y = "BKNG Price") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title
# Display the plot
p1+p2+plot_layout(ncol=1)
```





Observing the stock valuation through a seasonal lens can indicate recurring tendencies at different points within the year, potentially linked to the company's scheduled activities, financial disclosures, or consumer purchasing patterns that shift with the seasons. The dimensions of the boxes and the length of the whiskers provide insights into the stability or instability of stock prices during these periods.For RTX, the median price seems to be relatively stable over the years, with a slight increase in 2023. Spring and fall show less variability in price compared to summer and winter.For BKNG, it can be observed that there is significant variability in prices, with the largest ranges in the summer and winter.The winter of 2020 and 2023 shows a notably higher price range compared to other seasons, suggesting higher volatility or larger price swings.

To effectively forecast stock valuations, relying exclusively on the closing price proves inadequate. It's essential to delve into a broader range of indicators derived from the price history to enhance the model's learning capabilities. Since stock values are sequential in nature, we can extract various metrics from the historical pricing. In our analysis, we've incorporated a suite of technical indicators such as the moving average (MA), Moving Average Convergence Divergence (MACD), Bollinger Bands, 20-Day Standard Deviation (20sd), and Momentum.

The Moving Average[1] streamlines price data to create a continuously updated mean price, mitigating volatility and random price movements. This averaging offers a more transparent view of the pricing trend, with ascending values signaling an uptrend and descending values suggesting a downtrend. For our purposes, we've selected a 7-day span for short-term analysis and a 21-day span for a medium-term outlook. It can be calculated by:

The MACD[2] is a tool for gauging the momentum and trend changes of stock prices. A crossover of the MACD line above the signal line flags a potential buying opportunity, indicating a bullish trend. On the flip side, a crossover below hints at a bearish trend, possibly signaling a selling point.

Bollinger Bands[3] are utilized to assess market volatility, particularly for pinpointing overbought or oversold scenarios. Constriction of these bands can often herald an imminent sharp price movement. The upper band typically denotes an overbought market condition—a cue for potential selling, while proximity to the lower band suggests an oversold condition, which could be a buying opportunity.

Lastly, the Momentum indicator is instrumental in determining the velocity of price changes by comparing current prices to past values. This metric can often lead the way in forecasting trend changes, with positive values indicating a potential upward trajectory and negative values warning of a potential downtrend.

The following data frame represents a subset original dataset with extra features.

```r
# Define the function to calculate technical indicators
get_technical_indicators <- function(dataset) {
  # Create 7 and 21 days Moving Average
  dataset$ma7 <- rollmean(dataset[,2], k = 7, fill = NA)
  dataset$ma21 <- rollmean(dataset[,2], k = 21, fill = NA)

  # Create MACD
  dataset$'26ema' <- EMA(dataset[,2], n = 26)
  dataset$'12ema' <- EMA(dataset[,2], n = 12)
  dataset$MACD <- dataset$'12ema' - dataset$'26ema'

  # Create Bollinger Bands
  dataset$'20sd' <- rollapply(dataset[,2], width = 21, FUN = sd, fill = NA)
  dataset$upper_band <- dataset$ma21 + (dataset$'20sd' * 2)
  dataset$lower_band <- dataset$ma21 - (dataset$'20sd' * 2)

  # Create Exponential moving average
  #dataset$ema <- EMA(dataset$JNJ, n = 1)
  #n1 <- com_to_n(0.5)
  #dataset$ema <- EMA(dataset$JNJ, n = n1)

  # Create Momentum
  dataset$momentum <- dataset[,2] - 1
  dataset$log_momentum <- log(dataset$momentum)
  return(dataset)
}

RTX_data_extra <- get_technical_indicators(RTX_data)
RTX_data_extra <- na.omit(RTX_data_extra)
head(RTX_data_extra)
```

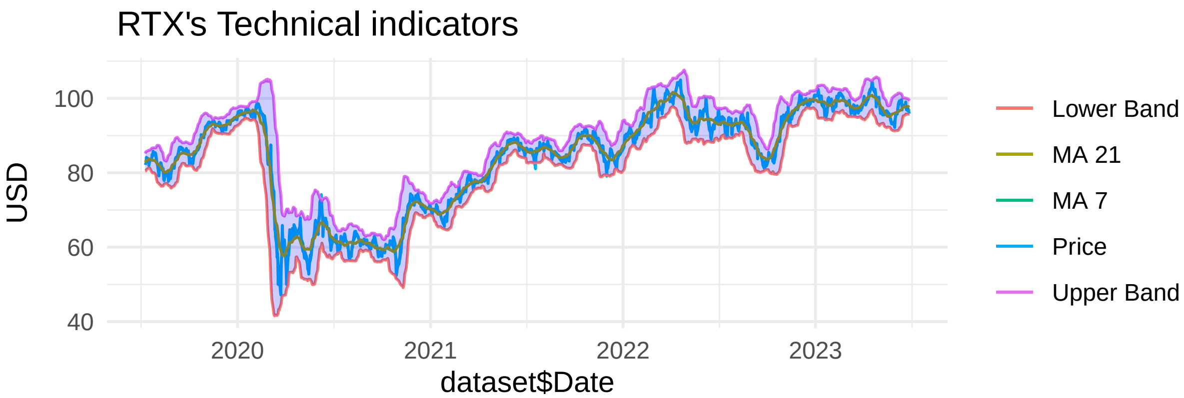
```
##          Date      RTX      ma7     ma21    26ema    12ema      MACD      20sd
## 26 2000-08-17 19.66646 19.45855 19.39534 18.70393 19.04117 0.3372326 0.5453017
## 27 2000-08-18 19.35179 19.60043 19.46839 18.75192 19.08896 0.3370317 0.4376647
## 28 2000-08-21 19.98112 19.74372 19.50210 18.84298 19.22621 0.3832361 0.3891913
## 29 2000-08-22 20.05979 19.86172 19.54424 18.93311 19.35445 0.4213441 0.4019200
## 30 2000-08-23 20.16795 19.92072 19.57796 19.02458 19.47961 0.4550278 0.4076936
## 31 2000-08-24 19.90245 19.96848 19.59294 19.08961 19.54466 0.4550535 0.3970191
##    upper_band lower_band momentum log_momentum
## 26   20.48594   18.30474 18.66646     2.926728
## 27   20.34372   18.59306 18.35179     2.909727
## 28   20.28048   18.72372 18.98112     2.943445
## 29   20.34808   18.74040 19.05979     2.947581
## 30   20.39334   18.76257 19.16795     2.953240
## 31   20.38698   18.79890 18.90245     2.939292
```

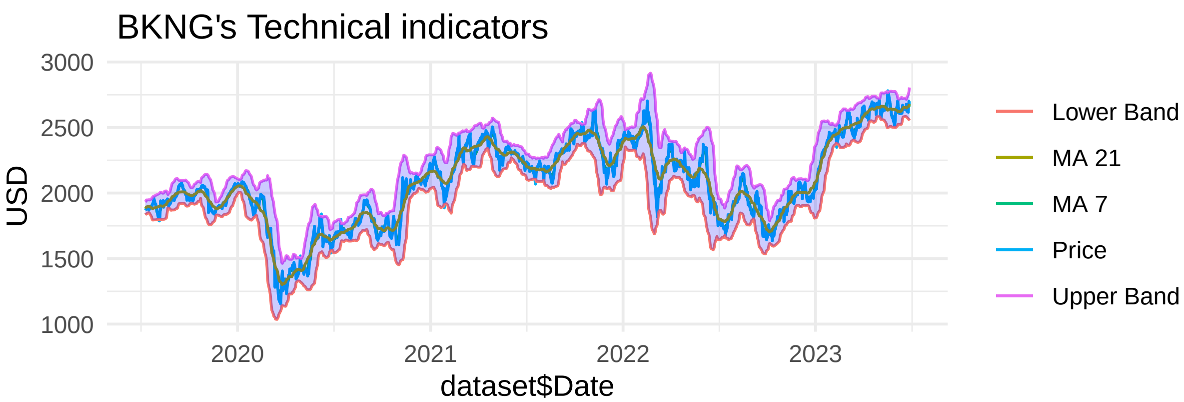
```
BKNG_data_extra <- get_technical_indicators(BKNG_data)
BKNG_data_extra <- na.omit(BKNG_data_extra)
#head(BKNG_data_extra)
```

These metrics play a crucial role in analyzing stock price movements and enhancing the dataset's features from a singular to an eleven-dimensional form. This expansion significantly aids the supervised learning model in comprehending the trends, correlations, and various interactions within the time series data.

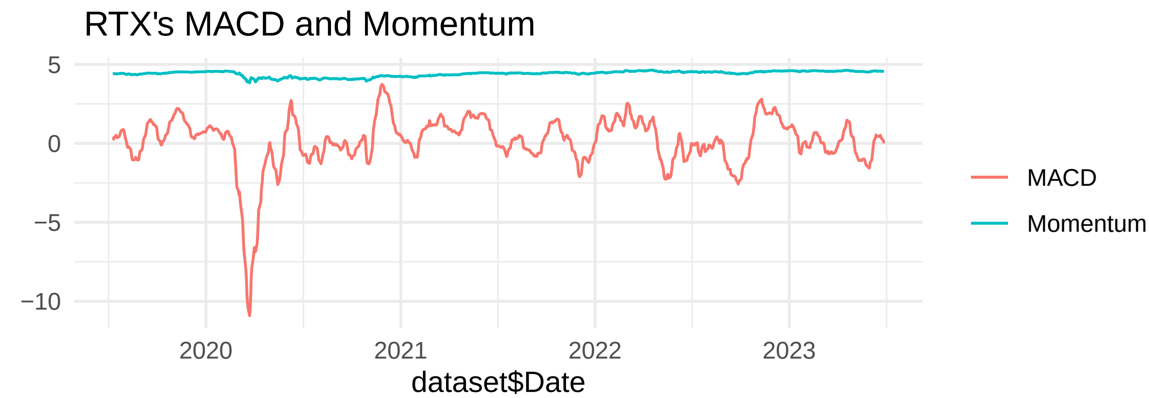
The following visuals display the indicators, providing a richer and clearer comprehension of the data.

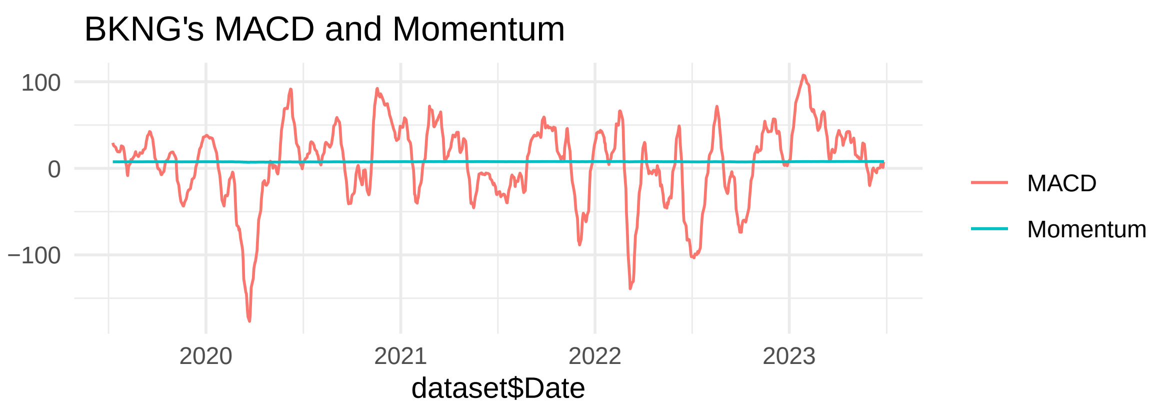
```
plot_technical_indicators <- function(dataset, last_days) {
  # Select the last 'last_days' of data
  dataset <- tail(dataset, last_days)

  # Create the first plot with MA7, JNJ Closing Price, MA21, and Bollinger Bands
  p1 <- ggplot(dataset, aes(x = dataset$Date)) +
    geom_line(aes(y = ma7, colour = "MA 7")) +
    geom_line(aes(y = dataset[,2], colour = "Price")) +
    geom_line(aes(y = ma21, colour = "MA 21")) +
    geom_line(aes(y = upper_band, colour = "Upper Band")) +
    geom_line(aes(y = lower_band, colour = "Lower Band")) +
    geom_ribbon(aes(ymin = lower_band, ymax = upper_band), fill = "blue", alpha = 0.2) +
    labs(title = sprintf(" %s's Technical indicators",names(dataset)[2]), y = "USD") +
    theme_minimal() +
    theme(legend.title = element_blank())
  return(p1)

}
p1=plot_technical_indicators(RTX_data_extra, 1000)
p2=plot_technical_indicators(BKNG_data_extra, 1000)
p1+p2+plot_layout(ncol=1)
```






```
plot_technical_indicators_MACD <- function(dataset, last_days){
  dataset <- tail(dataset, last_days)
    # Create the second plot with MACD and Momentum
  p2 <- ggplot(dataset, aes(x = dataset$Date)) +
    geom_line(aes(y = MACD, colour = "MACD")) +
    geom_line(aes(y = log_momentum, colour = "Momentum")) +
    labs(title = sprintf(" %s's MACD and Momentum",names(dataset)[2]), y = "") +
    theme_minimal() +
    theme(legend.title = element_blank())
  return(p2)
}
p1=plot_technical_indicators_MACD(RTX_data_extra, 1000)
p2=plot_technical_indicators_MACD(BKNG_data_extra, 1000)
p1+p2+plot_layout(ncol=1)
```



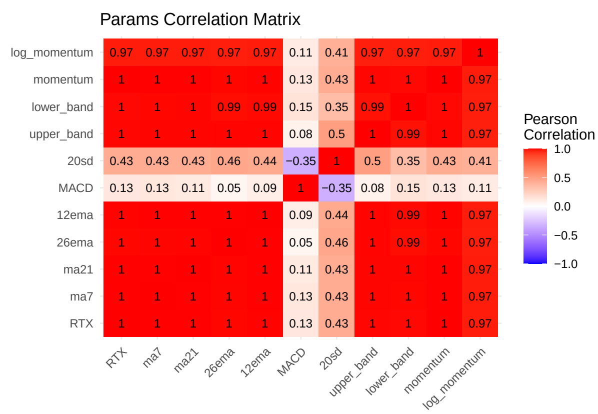


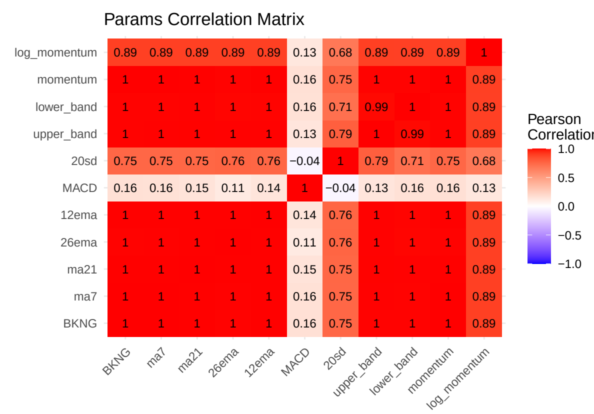
In the correlation heatmap provided in the following figure, the interrelations between several financial metrics are quantified using Pearson correlation coefficients. Notably, the heatmap reveals a strong positive correlation amongst the majority of indicators, particularly the moving averages and exponential moving averages. Their correlation values approach or reach 1, denoting a near-identical directional movement, which aligns with expectations given their derivation from the same underlying JNJ stock price data.

In contrast, the 20-day standard deviation indicator shows a less pronounced correlation with the MACD indicator, highlighting that volatility assessments and momentum indicators may not exhibit parallel behaviors. Additionally, the momentum and its logarithmic variant share a correlation of unity, underscoring the intrinsic mathematical relationship between a value and its logarithm.

```
RTX_data_extra_no_date <- RTX_data_extra %>%   select( -Date)
BKNG_data_extra_no_date <- BKNG_data_extra %>%   select( -Date)

# Calculate the correlation matrix
cor_matrix1 <- cor(RTX_data_extra_no_date, use = "complete.obs")
cor_matrix2 <- cor(BKNG_data_extra_no_date, use = "complete.obs")

# Set up the plot size
options(repr.plot.width = 28, repr.plot.height = 28)

library(reshape2)
melted_cor_matrix <- melt(cor_matrix1)

p1=ggplot(melted_cor_matrix, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
```

```r
                              midpoint = 0, limit = c(-1,1), space = "Lab",
                              name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = '', y = '', title = 'Params Correlation Matrix') +
  geom_text(aes(label = round(value, 2)), size = 3)
melted_cor_matrix <- melt(cor_matrix2)

p2=ggplot(melted_cor_matrix, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                              midpoint = 0, limit = c(-1,1), space = "Lab",
                              name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = '', y = '', title = 'Params Correlation Matrix') +
  geom_text(aes(label = round(value, 2)), size = 3)
p1;p2
```





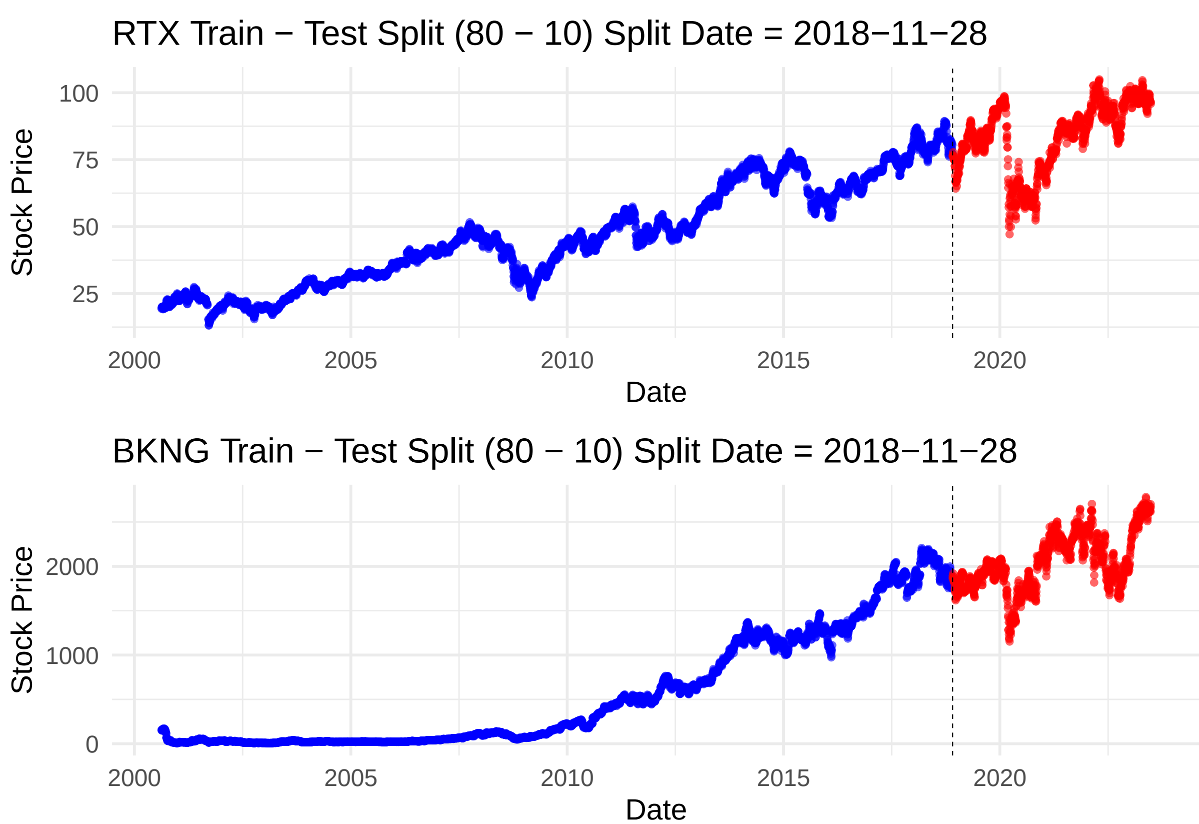
The subsequent phase in preparing the data involves partitioning the dataset into two subsets: a training set and a test set. We allocate the initial 80% of the dataset for training purposes, reserving the latter 20% as the test set. The split date index is 2018-11-28. The 'Date' column will be excluded from the training subset to avoid introducing noise into the model, as it is unsuitable for normalization through min-max scaling. Instead, we will construct an array comprising sequences of 60 days' worth of prices, with each sequence serving as an individual input for the LSTM model. The upcoming visual will depict how the dataset is divided between the training and testing phases.

```r
split_index <- floor(0.8 * nrow(RTX_data_extra))
split_date <- RTX_data_extra$Date[split_index]

print(split_date)
```

```
## [1] "2018-11-28"
```

```r
# Split the data into training and testing sets
data_training1 <- RTX_data_extra[1:split_index, ]
data_testing1 <- RTX_data_extra[(split_index + 1):nrow(RTX_data_extra), ]
data_training2 <- BKNG_data_extra[1:split_index, ]
```

```
data_testing2 <- BKNG_data_extra[(split_index + 1):nrow(BKNG_data_extra), ]
p1 <- ggplot() +
  geom_point(data = data_training1, aes(x =data_training1$Date , y = data_training1$RTX), color = 'blue
  geom_point(data = data_testing1, aes(x = data_testing1$Date, y = data_testing1$RTX), color = 'red', s
  geom_vline(xintercept = as.numeric(split_date), linetype = "dashed", color = "black", size = 0.2) +
  labs(title = 'RTX Train - Test Split (80 - 10) Split Date = 2018-11-28', x = 'Date', y = 'Stock Price
  theme_minimal() +
  theme(legend.position = "none")
p2 <- ggplot() +
  geom_point(data = data_training1, aes(x =data_training2$Date , y = data_training2$BKNG), color = 'blue
  geom_point(data = data_testing1, aes(x = data_testing2$Date, y = data_testing2$BKNG), color = 'red', s
  geom_vline(xintercept = as.numeric(split_date), linetype = "dashed", color = "black", size = 0.2) +
  labs(title = 'BKNG Train - Test Split (80 - 10) Split Date = 2018-11-28', x = 'Date', y = 'Stock Price
  theme_minimal() +
  theme(legend.position = "none")

# Display the plot
p1+p2+plot_layout(ncol=1)
```



After partitioning of the data, we normalized the data with a customized Min-Max scaling function. This step is critical because normalization ensure that no individual feature dominates the model's learning process due to its numerical range. In the context of LSTM which uses gradient-based optimization techniques, normalized data help in maintaining stable gradients, preventing issues like exploding or vanishing gradients, which are particularly problematic in LSTMs due to their recurrent nature.

```r
# Define a Min-Max scaling function
range01 <- function(x) (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
data_training1 <- data_training1 %>% select(-Date)
data_testing1 <- data_testing1 %>% select(-Date)
data_training2 <- data_training2 %>% select(-Date)
data_testing2 <- data_testing2 %>% select(-Date)
# Apply the scaling to each column
data_training_scaled1 <- as.data.frame(lapply(data_training1, range01))
data_training_scaled2 <- as.data.frame(lapply(data_training2, range01))

# Print the shape of the scaled data
cat("The scaled training data has", nrow(data_training_scaled1), "samples and", ncol(data_training_scal
```

```
## The scaled training data has 4600 samples and 11 features.
```

```r
# View the scaled data
#print(data_training_scaled)
```

In terms of data structure, the training feature set has been shaped into a three-dimensional array with dimensions [4540, 60, 11]. This indicates a compilation of 4540 instances, each containing a sequence of 60 time steps, with 11 distinct features. Correspondingly, the training target set comprises 4540 data points, each representing the output label for the LSTM to predict.

```r
X_train1 <- list()
y_train1 <- vector()
X_train2 <- list()
y_train2 <- vector()
# Loop to create sequences of 60 days for X_train and the next day's value for y_train
for (i in 61:nrow(data_training_scaled1)) {
  X_train1[[i - 60]] <- data_training_scaled1[(i-60):(i-1), ]
  y_train1[i - 60] <- data_training_scaled1[i, 1]  # Assuming the target variable is the first column
  X_train2[[i - 60]] <- data_training_scaled2[(i-60):(i-1), ]
  y_train2[i - 60] <- data_training_scaled2[i, 1]
}

# Convert the list to an array-like structure
X_train1 <- array(unlist(X_train1), dim = c(length(X_train1), 60, ncol(data_training_scaled1)))
y_train1 <- as.numeric(y_train1)  # Convert y_train to a numeric vector
X_train2 <- array(unlist(X_train2), dim = c(length(X_train2), 60, ncol(data_training_scaled2)))
y_train2 <- as.numeric(y_train2)
# Print the shapes of X_train and y_train
cat("X_train shape:", dim(X_train2), "\n")
```

```
## X_train shape: 4540 60 11
```

```r
cat("y_train shape:", length(y_train2), "\n")
```

```
## y_train shape: 4540
```

5.2 Model Structure

For the supervised learning model, we implemented a RNN model consists of a sequential LSTM network. The first layer is an LSTM layer with 50 units. The activation function is ‘relu’, which stands for rectified linear unit. Following the first LSTM layer, there’s a dropout layer with a dropout rate of 0.2 (20%). Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of the input units to 0 during training.The second LSTM layer with 60 units is followed by the second Dropout Layer with a dropout rate of 0.3. The third and forth LSTM Layer has 80 and 120 units respectivly, and followed by the corresponding dropout layer with dropout rate of 0.4 and 0.5. The model concludes with a dense layer with a single unit, which represents the final price prediction.

```r
library(reticulate)
#reticulate::install_python(version = "3.9:latest")
# Create a virtual environment named 'r-reticulate' in the default location
#virtualenv_create(envname = "r-reticulate")



#install package needed for this project
#py_install(packages = c("tensorflow", "keras"), envname = "r-reticulate")

# Use the newly created virtual environment
use_virtualenv("r-reticulate", required = TRUE)
```

```r
# Initialize the RNN model
model1 <- keras_model_sequential()

# Adding the first LSTM layer and some Dropout regularization
model1 %>%
  layer_lstm(units = 50, activation = 'relu', return_sequences = TRUE, input_shape = c(dim(X_train1)[2]
  layer_dropout(0.2)

# Adding a second LSTM layer and some Dropout regularization
model1 %>%
  layer_lstm(units = 60, activation = 'relu', return_sequences = TRUE) %>%
  layer_dropout(0.3)

# Adding a third LSTM layer and some Dropout regularization
model1 %>%
  layer_lstm(units = 80, activation = 'relu', return_sequences = TRUE) %>%
  layer_dropout(0.4)

# Adding a fourth LSTM layer and some Dropout regularization
model1 %>%
  layer_lstm(units = 120, activation = 'relu') %>%
  layer_dropout(0.5)

# Adding the output layer
model1 %>%
  layer_dense(units = 1)

# Compiling the RNN
model1 %>% compile(
  optimizer = 'adam',
```

```r
  loss = 'mean_squared_error'
)

# Summary of the model to confirm the architecture
summary(model1)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  lstm (LSTM)                        (None, 60, 50)                    12400
##  dropout (Dropout)                  (None, 60, 50)                    0
##  lstm_1 (LSTM)                      (None, 60, 60)                    26640
##  dropout_1 (Dropout)                (None, 60, 60)                    0
##  lstm_2 (LSTM)                      (None, 60, 80)                    45120
##  dropout_2 (Dropout)                (None, 60, 80)                    0
##  lstm_3 (LSTM)                      (None, 120)                       96480
##  dropout_3 (Dropout)                (None, 120)                       0
##  dense (Dense)                      (None, 1)                         121
## ================================================================================
## Total params: 180761 (706.10 KB)
## Trainable params: 180761 (706.10 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```

```r
# Initialize the RNN model
model2 <- keras_model_sequential()

# Adding the first LSTM layer and some Dropout regularization
model2 %>%
  layer_lstm(units = 50, activation = 'relu', return_sequences = TRUE, input_shape = c(dim(X_train2)[2]
  layer_dropout(0.2)

# Adding a second LSTM layer and some Dropout regularization
model2 %>%
  layer_lstm(units = 60, activation = 'relu', return_sequences = TRUE) %>%
  layer_dropout(0.3)

# Adding a third LSTM layer and some Dropout regularization
model2 %>%
  layer_lstm(units = 80, activation = 'relu', return_sequences = TRUE) %>%
  layer_dropout(0.4)

# Adding a fourth LSTM layer and some Dropout regularization
model2 %>%
  layer_lstm(units = 120, activation = 'relu') %>%
  layer_dropout(0.5)

# Adding the output layer
model2 %>%
  layer_dense(units = 1)

# Compiling the RNN
```

```r
model2 %>% compile(
  optimizer = 'adam',
  loss = 'mean_squared_error'
)

# Summary of the model to confirm the architecture
#summary(model2)
```

5.3 Model Training and Result

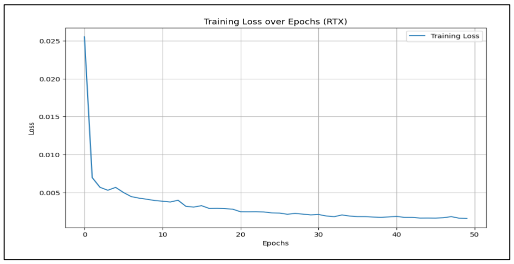
The model was developed and compiled within an R environment utilizing the reticulate package to bridge Python functionality. This approach introduces several potential constraints that might affect the model's performance. For example, floating-point operations are inherently prone to rounding errors due to their approximate nature in digital storage. The oneDNN library, accessed via the reticulate package, aims to enhance performance by potentially altering computational sequences. Such modifications, while boosting efficiency, might result in minor discrepancies in outcomes due to the intrinsic properties of floating-point math. These slight numerical variances can pose significant challenges in ensuring result reproducibility, a critical factor in research or scenarios requiring precise outcome verification and model fine-tuning.

Furthermore, Python's comprehensive ecosystem for deep learning typically yields more effective model training outcomes. Consider the Min-Max scaler, which is inherently more robust in Python, whereas in R, a custom implementation is required. Python's environment generally delivers superior performance for training complex models, backed by libraries optimized for speed and GPU acceleration capabilities. For instance, during the initial phase of model training, Python demonstrated a lower initial validation loss compared to its R counterpart, indicating a more efficient start. Despite multiple attempts to achieve satisfactory results with the R model, the outcomes weren't as promising. Consequently, the decision was made to train the model in Python, utilizing identical settings, to harness its optimized computational environment and robust deep learning infrastructure.

The model was trained for 50 epochs using a batch size of 64. Subsequently, it was evaluated on the test dataset, which included an extra 60 instances from the end of the training data to initialize the model. The following figure shows the loss(Mean Squared Error) during training in Python.

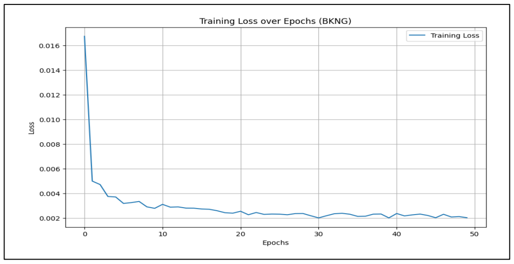
```r
#install.packages("png")
library(png)
img1 <- readPNG('loss_rtx.png')
img2 <- readPNG('loss_BKNG.png')


# Plot the first image
plot(0:1, 0:1, type='n', xlab='', ylab='', xaxt='n', yaxt='n', main='RTX Loss')
rasterImage(img1, 0, 0, 1, 1)
```

RTX Loss



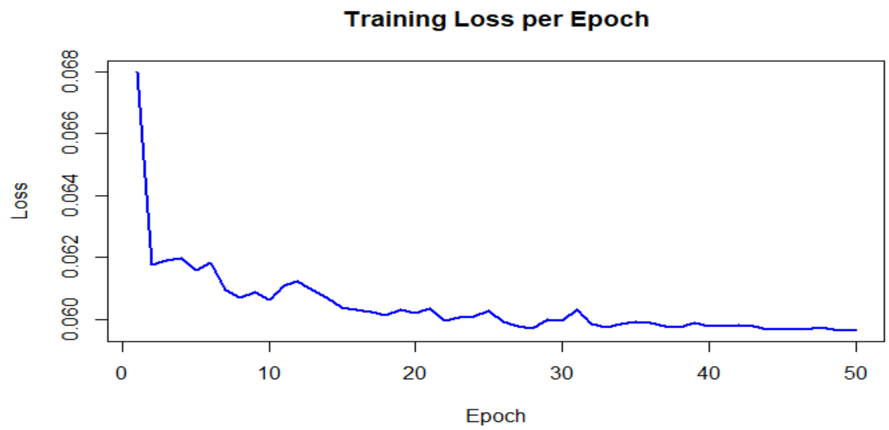

```r
# Plot the second image
plot(0:1, 0:1, type='n', xlab='', ylab='', xaxt='n', yaxt='n', main='BKNG Loss')
rasterImage(img2, 0, 0, 1, 1)
```

BKNG Loss



For comparsion, the following figure shows the training loss in R, with the same setting.

```r
#install.packages("png")
library(png)
img <- readPNG('loss_r.png')
plot.new()
rasterImage(img,0,0,1,1)
```



In the following displayed graph, the performance of the LSTM-based RNN model is visualized, showcasing its proficiency in tracking the stock's directional trends. The blue line is the predicted price and the red line represents the actual price. While there are instances of divergence from the actual stock figures, notably at the high and low points, the model generally mirrors the true course of the stock prices. This alignment suggests that the model has effectively internalized historical pricing patterns, which bodes well for its potential in forecasting forthcoming stock movements or analyzing market behaviors.

The model's consistent capture of the overall stock price patterns of RTX and BKNG is encouraging; however, it also brings to light the intricate nature of the financial markets, swayed by a multitude of unpredictable factors such as international political scenarios and investor sentiment.

Notably, the model smooths over the predictions, indicating a propensity towards recognizing long-standing trends rather than short-lived market volatilities. Enhancing the model's sensitivity to immediate market transitions is a key area for further development, given the significance of such adaptability in trading strategies.

Future enhancements should focus on the integration of a wider array of predictive factors, including trading activity levels, sentiment from financial reporting, and macroeconomic indicators, expanding beyond the primary dependence on closing prices. Additionally, experimenting with hybrid neural network configurations that combine LSTM and advanced architectures like Transformer models could yield greater insights into effective stock price prediction methodologies.

```
#install.packages("png")
library(png)
img1 <- readPNG('result_rtx.png')
img2 <- readPNG('result_bkng.png')
```

```r
# Plot the first image
plot(0:1, 0:1, type='n', xlab='', ylab='', xaxt='n', yaxt='n', main='RTX Prrdiction Result')
rasterImage(img1, 0, 0, 1, 1)
```

RTX Prrdiction Result


```r
# Plot the second image
plot(0:1, 0:1, type='n', xlab='', ylab='', xaxt='n', yaxt='n', main='BKNG Prrdiction Result')
rasterImage(img2, 0, 0, 1, 1)
```

BKNG Prrdiction Result

6 Reference

1. Adam Hayes. Dotcom Bubble Definition. ***Investopedia***, Economics(2023).\Available online: https://www.investopedia.com/terms/d/dotcom-bubble.asp.

2. Bahl, J. Study of Predictive Power of Moving Averages as a Tool of Technical Analysis. ***Journal for Studies in Management and Planning***, 1(2).(2015). \Available online: https://ssrn.com/abstract=3037580.

3. Chio, P. T. A comparative study of the MACD-base trading strategies: evidence from the US stock market. ***arXiv preprint***(2022).arXiv:2206.12282.

4. Javier E.Darid. BREXIT Brexit cost investors $2 trillion, the worst one day drop ever ***CNBC***, (2016). \Available online: https://www.cnbc.com/2016/06/26/brexit-cost-investors-2-trillion-the-worst-one-day-drop-ever.html

5. McNeil, A. J., Frey, R., & Embrechts, P. Quantitative risk management: Concepts, techniques and tools. ***Princeton University Press***. (2015).

6. Noel Randewich. Wall Street chalks up biggest gain in four years. ***Reuters*** Business(2015). \Available online: https://www.reuters.com/article/us-markets-stocks-usa-idUSKCN0QV12320150826/

7. Pinakin, S. N., & Manubhai, P. T. A comparative study on technical analysis by Bollinger Band and RSI. ***International Journal in Management & Social Science***, 3(6),(2015). 234-251.

8. Quantile-Quantile Plots \Available online: https://www.ucd.ie/ecomodel/Resources/QQplots_WebVersion.html.

9. Steve Hargreaves. Investors lose a trillion dollars in one day. ***CNN***, Money(2011).\Available online: https://money.cnn.com/2011/08/08/markets/stock-market-loss/index.htm