stat5010 Final Project

Chanthrika Palanisamy

STOCK PRICE pREDICTION: AFTER RUSSIA - UKRAINE CONFLICT

##Experts says oil prices hiked were other sectors like health care, Utilities followed their usual trend with a little drop. Lets see howfar it is legit!

INTRODUCTION

###Stock prices are lower and energy prices are higher, in response to Russia's invasion of Ukraine. The U.S. stock market opened sharply lower, while crude oil prices topped $100 a barrel. RACHEL MARTIN, HOST: Russia's invasion of Ukraine has roiled financial markets around the world and sent energy prices soaring.

Figure 1: Caption for the picture.

In this project I considered 7 sectors,

#1. Energy Sector: Example took to represent the sector is Phillips66(PSX) #2. Technology Sector: Example took to represent the sector is Apple(AAPL) #3. Utilities Sector: Example took to represent the sector is Xcel Energy(XEL) #4. Health Care Sector: Example took to represent the sector is CVS Pharmacy(CVS) #5. Communication Services Sector: Example took to represent the sector is Alphabet Inc Class A(GOOG) #6. Consumer Discretionary Sector: Example took to represent the sector is Tesla(TSLA) #7. Consumer Staples Sector: Example took to represent the sector is Cocacola(KO)

1. DATA COLLECTION:

###The project import data and built-in functions from "quantmod' library. Quantmod library import stock data from yahoo finance that were used in the report. The variables of the data:

Closing Price is the closing price of the day when the market closes at 4:00PM
Opening Price is the opening price of the day when the market opens at 9:30AM
High and Low Price are the highest and lowest price traded during the day, respectively
Volume which are the amount of shares being bought and sold during the day
Adjusted Price is the price reflect that stock's value after accounting for any corporate actions such as stock splits, dividends / distributions and rights offerings

```
getSymbols("XEL",from="2009-03-06",to="2022-03-25", src='yahoo')
```

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

```
## [1] "XEL"
```

```
getSymbols("AAPL",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "AAPL"
```

```
getSymbols("GOOG",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "GOOG"
```

```
getSymbols("AMT",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "AMT"
```

```
getSymbols("CVS",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "CVS"
```

```
getSymbols("KO",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "KO"
```

```
getSymbols("PSX",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "PSX"
```

```
getSymbols("TSLA",from="2009-03-06",to="2022-03-25", src='yahoo')
```

```
## [1] "TSLA"
```

```
XEL = data.frame(Date=index(XEL),coredata(XEL))
XEL$Date <- sapply(XEL$Date, as.character)
names(XEL) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
XEL <- XEL%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
AAPL = data.frame(Date=index(AAPL),coredata(AAPL))
AAPL$Date <- sapply(AAPL$Date, as.character)
AAPL<-AAPL%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
names(AAPL) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
AMT = data.frame(Date=index(AMT),coredata(AMT))
AMT$Date <- sapply(AMT$Date, as.character)
AMT <- AMT%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
names(AMT) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
CVS = data.frame(Date=index(CVS),coredata(CVS))
CVS$Date <- sapply(CVS$Date, as.character)
CVS<-CVS%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
names(CVS) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
KO = data.frame(Date=index(KO),coredata(KO))
KO$Date <- sapply(KO$Date, as.character)
KO<-KO%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
names(KO) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
PSX = data.frame(Date=index(PSX),coredata(PSX))
PSX$Date <- sapply(PSX$Date, as.character)
PSX <- PSX%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
names(PSX) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
TSLA = data.frame(Date=index(TSLA),coredata(TSLA))
TSLA$Date <- sapply(TSLA$Date, as.character)
TSLA <- TSLA%>%mutate(Date = as.Date(Date)) %>%
  complete(Date = seq.Date(min(Date), max(Date), by="day"))
names(TSLA) <- c('Date', 'Open','High','Low','Close','Volume','Adj.Close')
```

#2. DATA PREPROCESSING:

Addition variables are *TrueRange*, *Company_Name* and *Growth*. *TrueRange* is High minus Low. This tracks the volatility of stock.This an important variables because, day trader relies on the price variation to

make profits, a stock with high TrueRange compared to the price means a better opportunity to make profit. *Growth* is the indicator variables that tracks if closing price is higher than the opening price. *Company_Name* is an indicator variable which makes it easy for us to group the data and manipulate the data with respect to the company name.

```
XEL <- XEL%>%mutate(Company_Name = "XEL")
AAPL <- AAPL%>%mutate(Company_Name="AAPL")
CVS <- CVS%>%mutate(Company_Name ="CVS")
KO <- KO%>%mutate(Company_Name ="KO")
PSX <- PSX%>%mutate(Company_Name = "PSX")
TSLA <- TSLA%>%mutate(Company_Name ="TSLA")
# XEL = XEL%>% mutate(Day = day(Date))
Growth.Function = function(net){
  ifelse(net>= 0, 1,0)
}
XEL['Growth'] <- Growth.Function(XEL$Close-XEL$Open)

XEL <- XEL%>% mutate(TrueRange = High-Low)
head(XEL)
```

```
## # A tibble: 6 x 10
##   Date        Open  High   Low Close  Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>     <dbl> <chr>          <dbl>
## 1 2009-03-06  16.6  17.0  16.2  16.6 4651400      10.3 XEL                0
## 2 2009-03-07  NA    NA    NA    NA         NA      NA   XEL               NA
## 3 2009-03-08  NA    NA    NA    NA         NA      NA   XEL               NA
## 4 2009-03-09  16.4  16.5  16.1  16.2 2951500      10.1 XEL                0
## 5 2009-03-10  16.5  16.6  16.3  16.6 4274400      10.3 XEL                1
## 6 2009-03-11  16.0  16.9  16.0  16.6 3410500      10.4 XEL                1
## # ... with 1 more variable: TrueRange <dbl>
```

```
XEL['Growth'] <-  Growth.Function(XEL$Close-XEL$Open)

PSX = PSX%>% mutate(TrueRange = High-Low)
head(PSX)
```

```
## # A tibble: 6 x 9
##   Date        Open  High   Low Close  Volume Adj.Close Company_Name TrueRange
##   <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>     <dbl> <chr>             <dbl>
## 1 2012-04-12  33.2  34.2  32    34    654700      24.7 PSX                2.21
## 2 2012-04-13  34    37.5  33.9  37   1685000      26.9 PSX                3.54
## 3 2012-04-14  NA    NA    NA    NA         NA      NA   PSX               NA
## 4 2012-04-15  NA    NA    NA    NA         NA      NA   PSX               NA
## 5 2012-04-16  37    37    36.0  36.5  550200      26.5 PSX                1.01
## 6 2012-04-17  36.4  36.8  36    36    179600      26.1 PSX                0.75
```

```
PSX['Growth'] <- Growth.Function(PSX$Close-PSX$Open)

PSX = PSX%>% mutate(TrueRange = High-Low)
head(PSX)
```

```
## # A tibble: 6 x 10
```

```
##   Date        Open  High   Low Close  Volume Adj.Close Company_Name TrueRange
##   <date>     <dbl> <dbl> <dbl> <dbl>   <dbl>     <dbl> <chr>             <dbl>
## 1 2012-04-12  33.2  34.2  32    34    654700      24.7 PSX                2.21
## 2 2012-04-13  34    37.5  33.9  37   1685000      26.9 PSX                3.54
## 3 2012-04-14  NA    NA    NA    NA        NA      NA   PSX               NA
## 4 2012-04-15  NA    NA    NA    NA        NA      NA   PSX               NA
## 5 2012-04-16  37    37    36.0  36.5  550200      26.5 PSX                1.01
## 6 2012-04-17  36.4  36.8  36    36    179600      26.1 PSX                0.75
## # ... with 1 more variable: Growth <dbl>
```

```
KO['Growth'] <- Growth.Function(KO$Close-KO$Open)

KO = KO%>% mutate(TrueRange = High-Low)
head(KO)
```

```
## # A tibble: 6 x 10
##   Date        Open  High   Low Close   Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>     <dbl> <chr>          <dbl>
## 1 2009-03-06  19.1  19.6  19.0  19.5 33002400      12.9 KO                 1
## 2 2009-03-07  NA    NA    NA    NA         NA      NA   KO                NA
## 3 2009-03-08  NA    NA    NA    NA         NA      NA   KO                NA
## 4 2009-03-09  19.7  19.9  19.4  19.4 35127400      12.8 KO                 0
## 5 2009-03-10  19.8  19.8  19.3  19.6 35181400      12.9 KO                 0
## 6 2009-03-11  19.6  20.0  19.5  19.8 27254000      13.2 KO                 1
## # ... with 1 more variable: TrueRange <dbl>
```

```
AAPL['Growth'] <- Growth.Function(AAPL$Close-AAPL$Open)

AAPL = AAPL%>% mutate(TrueRange = High-Low)
head(AAPL)
```

```
## # A tibble: 6 x 10
##   Date        Open  High   Low Close     Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>      <dbl>     <dbl> <chr>          <dbl>
## 1 2009-03-06  3.16  3.16  2.94  3.05 1011147200      2.61 AAPL               0
## 2 2009-03-07 NA    NA    NA    NA            NA     NA    AAPL              NA
## 3 2009-03-08 NA    NA    NA    NA            NA     NA    AAPL              NA
## 4 2009-03-09  3.01  3.13  2.95  2.97  698297600      2.54 AAPL               0
## 5 2009-03-10  3.03  3.18  3.01  3.17  844258800      2.71 AAPL               1
## 6 2009-03-11  3.21  3.36  3.20  3.31  846372800      2.83 AAPL               1
## # ... with 1 more variable: TrueRange <dbl>
```

```
CVS['Growth'] <- Growth.Function(CVS$Close-CVS$Open)

CVS = CVS%>% mutate(TrueRange = High-Low)
head(CVS)
```

```
## # A tibble: 6 x 10
##   Date        Open  High   Low Close   Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>     <dbl> <chr>          <dbl>
## 1 2009-03-06  24.8  25.1  23.8  24.5 11878800      18.8 CVS                0
## 2 2009-03-07  NA    NA    NA    NA         NA      NA   CVS               NA
```

```
## 3 2009-03-08  NA    NA    NA    NA         NA      NA   CVS              NA
## 4 2009-03-09  24.2  24.6  23.7  24.0  9123000      18.4 CVS               0
## 5 2009-03-10  24.3  25.0  24.0  24.9 15470500      19.2 CVS               1
## 6 2009-03-11  25.0  25.8  25.0  25.2 11612800      19.4 CVS               1
## # ... with 1 more variable: TrueRange <dbl>
```

```
TSLA['Growth'] <- Growth.Function(TSLA$Close-TSLA$Open)

TSLA = TSLA%>% mutate(TrueRange = High-Low)
head(TSLA)
```

```
## # A tibble: 6 x 10
##   Date        Open  High   Low Close   Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>     <dbl> <chr>         <dbl>
## 1 2010-06-29  3.8   5     3.51  4.78 93831500      4.78 TSLA              1
## 2 2010-06-30  5.16  6.08  4.66  4.77 85935500      4.77 TSLA              0
## 3 2010-07-01  5     5.18  4.05  4.39 41094000      4.39 TSLA              0
## 4 2010-07-02  4.6   4.62  3.74  3.84 25699000      3.84 TSLA              0
## 5 2010-07-03 NA    NA    NA    NA          NA      NA    TSLA             NA
## 6 2010-07-04 NA    NA    NA    NA          NA      NA    TSLA             NA
## # ... with 1 more variable: TrueRange <dbl>
```

#2.1 Spliting Train and Test datasets

The time series data (usually) exhibits strong serial auto-correlation, so if we put the price for one day in the training set, and the next days price the test set they're a long way from independent and the test error is biased. Therefore we introduce a gap between the train and test dataset.

```
XEL_train <-head(XEL,round(0.65*nrow(XEL)))
XEL_test <- tail(XEL,round(0.30*nrow(XEL)))
AAPL_train <-head(AAPL,round(0.65*nrow(AAPL)))
AAPL_test <- tail(AAPL,round(0.30*nrow(AAPL)))
TSLA_train <-head(TSLA,round(0.65*nrow(AAPL)))
TSLA_test <- tail(TSLA,round(0.30*nrow(AAPL)))
CVS_train <-head(CVS,round(0.65*nrow(CVS)))
CVS_test <- tail(CVS,round(0.30*nrow(CVS)))
PSX_train <-head(PSX,round(0.65*nrow(PSX)))
PSX_test <- tail(PSX,round(0.30*nrow(PSX)))
KO_train <-head(KO,round(0.65*nrow(KO)))
KO_test <- tail(KO,round(0.30*nrow(KO)))
```

#As I wanted to predict the EPS (Earnings per Share) for the next quarter and EPS column is not available in Yahoo data source, I web scrapped the EPS from yahoo using 'rvest' library. The table available in the website is quarterly data. Though, I wasnt able to interpret th model to my statisfaction.

```
webscrap_Tesla <- read_html("https://www.macrotrends.net/stocks/charts/TSLA/tesla/eps-earnings-per-shar
tables_webscrap_Tesla <-webscrap_Tesla%>% html_table(fill = TRUE)
   webscrap_Tesla_table <- tables_webscrap_Tesla[[2]]
    names(webscrap_Tesla_table) <- c("Date", "EPS")
    webscrap_Tesla_table
```

```
## # A tibble: 51 x 2
##    Date       EPS
```

```
##    <chr>      <chr>
##  1 2021-12-31 $2.05
##  2 2021-09-30 $1.44
##  3 2021-06-30 $1.02
##  4 2021-03-31 $0.39
##  5 2020-12-31 $0.25
##  6 2020-09-30 $0.27
##  7 2020-06-30 $0.10
##  8 2020-03-31 $0.02
##  9 2019-12-31 $0.14
## 10 2019-09-30 $0.16
## # ... with 41 more rows
```

```
webscrap_Apple <- read_html("https://www.macrotrends.net/stocks/charts/AAPL/apple/eps-earnings-per-shar
tables_webscrap_Apple <-webscrap_Apple%>% html_table(fill = TRUE)
   webscrap_Apple_table <- tables_webscrap_Apple[[2]]
    names(webscrap_Apple_table) <- c("Date", "EPS")
    webscrap_Apple_table
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <chr>      <chr>
##  1 2021-12-31 $2.10
##  2 2021-09-30 $1.23
##  3 2021-06-30 $1.30
##  4 2021-03-31 $1.40
##  5 2020-12-31 $1.68
##  6 2020-09-30 $0.74
##  7 2020-06-30 $0.65
##  8 2020-03-31 $0.64
##  9 2019-12-31 $1.25
## 10 2019-09-30 $0.77
## # ... with 42 more rows
```

```
webscrap_Phillips66 <- read_html("https://www.macrotrends.net/stocks/charts/PSX/phillips-66/eps-earnings
tables_webscrap_Phillips66 <-webscrap_Phillips66%>% html_table(fill = TRUE)
   webscrap_Phillips66_table <- tables_webscrap_Phillips66[[2]]
    names(webscrap_Phillips66_table) <- c("Date", "EPS")
    webscrap_Phillips66_table
```

```
## # A tibble: 46 x 2
##    Date       EPS
##    <chr>      <chr>
##  1 2021-12-31 $2.88
##  2 2021-09-30 $0.91
##  3 2021-06-30 $0.66
##  4 2021-03-31 $-1.49
##  5 2020-12-31 $-1.25
##  6 2020-09-30 $-1.82
##  7 2020-06-30 $-0.33
##  8 2020-03-31 $-5.66
##  9 2019-12-31 $1.63
## 10 2019-09-30 $1.58
## # ... with 36 more rows
```

```
webscrap_CocaCola <- read_html("https://www.macrotrends.net/stocks/charts/KO/cocacola/eps-earnings-per-
tables_webscrap_CocaCola <-webscrap_CocaCola%>% html_table(fill = TRUE)
  webscrap_CocaCola_table <- tables_webscrap_CocaCola[[2]]
    names(webscrap_CocaCola_table) <- c("Date", "EPS")
    webscrap_CocaCola_table
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <chr>      <chr>
##  1 2021-12-31 $0.56
##  2 2021-09-30 $0.57
##  3 2021-06-30 $0.61
##  4 2021-03-31 $0.52
##  5 2020-12-31 $0.34
##  6 2020-09-30 $0.40
##  7 2020-06-30 $0.41
##  8 2020-03-31 $0.64
##  9 2019-12-31 $0.47
## 10 2019-09-30 $0.60
## # ... with 42 more rows
```

```
webscrap_CVS <- read_html("https://www.macrotrends.net/stocks/charts/CVS/cvs-health/eps-earnings-per-sha
tables_webscrap_CVS <-webscrap_CVS%>% html_table(fill = TRUE)
   webscrap_CVS_table <- tables_webscrap_CVS[[2]]
    names(webscrap_CVS_table) <- c("Date", "EPS")
    webscrap_CVS_table
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <chr>      <chr>
##  1 2021-12-31 $0.98
##  2 2021-09-30 $1.20
##  3 2021-06-30 $2.10
##  4 2021-03-31 $1.68
##  5 2020-12-31 $0.74
##  6 2020-09-30 $0.93
##  7 2020-06-30 $2.26
##  8 2020-03-31 $1.53
##  9 2019-12-31 $1.33
## 10 2019-09-30 $1.17
## # ... with 42 more rows
```

```
webscrap_Xcel <- read_html("https://www.macrotrends.net/stocks/charts/XEL/xcel-energy/eps-earnings-per-
tables_webscrap_Xcel <-webscrap_Xcel%>% html_table(fill = TRUE)
   webscrap_Xcel_table <- tables_webscrap_Xcel[[2]]
    names(webscrap_Xcel_table) <- c("Date", "EPS")
    webscrap_Xcel_table
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <chr>      <chr>
```

```
##  1 2021-12-31 $0.58
##  2 2021-09-30 $1.13
##  3 2021-06-30 $0.58
##  4 2021-03-31 $0.67
##  5 2020-12-31 $0.55
##  6 2020-09-30 $1.14
##  7 2020-06-30 $0.54
##  8 2020-03-31 $0.56
##  9 2019-12-31 $0.56
## 10 2019-09-30 $1.01
## # ... with 42 more rows
```

Formatting the columns to the original data frame column's format, because we need to merge it to the original dataset.

```
webscrap_Phillips66_table$Date <- (as.Date(webscrap_Phillips66_table$Date, format= "%Y-%m-%d"))
webscrap_Phillips66_table %>% arrange(ymd(webscrap_Phillips66_table$Date))
```

```
## # A tibble: 46 x 2
##    Date       EPS
##    <date>     <chr>
##  1 2009-12-31 $0.00
##  2 2010-12-31 $0.00
##  3 2011-03-31 $0.00
##  4 2011-06-30 $1.64
##  5 2011-09-30 $1.65
##  6 2011-12-31 $3.17
##  7 2012-03-31 $1.00
##  8 2012-06-30 $1.86
##  9 2012-09-30 $2.51
## 10 2012-12-31 $1.11
## # ... with 36 more rows
```

```
webscrap_Tesla_table$Date <- (as.Date(webscrap_Tesla_table$Date, format= "%Y-%m-%d"))
webscrap_Tesla_table %>% arrange(ymd(webscrap_Tesla_table$Date))
```

```
## # A tibble: 51 x 2
##    Date       EPS
##    <date>     <chr>
##  1 2009-06-30 $-0.31
##  2 2009-09-30 $-0.13
##  3 2009-12-31 $0.00
##  4 2010-03-31 $-0.81
##  5 2010-06-30 $-1.01
##  6 2010-09-30 $-0.08
##  7 2010-12-31 $1.28
##  8 2011-03-31 $-0.10
##  9 2011-06-30 $-0.12
## 10 2011-09-30 $-0.13
## # ... with 41 more rows
```

```
webscrap_Apple_table$Date <- (as.Date(webscrap_Apple_table$Date, format= "%Y-%m-%d"))
webscrap_Apple_table %>% arrange(ymd(webscrap_Apple_table$Date))
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <date>     <chr>
##  1 2009-03-31 $0.06
##  2 2009-06-30 $0.07
##  3 2009-09-30 $0.10
##  4 2009-12-31 $0.13
##  5 2010-03-31 $0.12
##  6 2010-06-30 $0.13
##  7 2010-09-30 $0.17
##  8 2010-12-31 $0.23
##  9 2011-03-31 $0.23
## 10 2011-06-30 $0.28
## # ... with 42 more rows
```

```
webscrap_CVS_table$Date <- (as.Date(webscrap_CVS_table$Date, format= "%Y-%m-%d"))
webscrap_CVS_table %>% arrange(ymd(webscrap_CVS_table$Date))
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <date>     <chr>
##  1 2009-03-31 $0.50
##  2 2009-06-30 $0.60
##  3 2009-09-30 $0.71
##  4 2009-12-31 $0.74
##  5 2010-03-31 $0.55
##  6 2010-06-30 $0.60
##  7 2010-09-30 $0.59
##  8 2010-12-31 $0.75
##  9 2011-03-31 $0.52
## 10 2011-06-30 $0.60
## # ... with 42 more rows
```

```
webscrap_CocaCola_table$Date <- (as.Date(webscrap_CocaCola_table$Date, format= "%Y-%m-%d"))
webscrap_CocaCola_table %>% arrange(ymd(webscrap_CocaCola_table$Date))
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <date>     <chr>
##  1 2009-03-31 $0.29
##  2 2009-06-30 $0.44
##  3 2009-09-30 $0.41
##  4 2009-12-31 $0.33
##  5 2010-03-31 $0.35
##  6 2010-06-30 $0.51
##  7 2010-09-30 $0.44
##  8 2010-12-31 $1.24
##  9 2011-03-31 $0.41
## 10 2011-06-30 $0.60
## # ... with 42 more rows
```

```
webscrap_Xcel_table$Date <- (as.Date(webscrap_Xcel_table$Date, format= "%Y-%m-%d"))
webscrap_Xcel_table %>% arrange(ymd(webscrap_Xcel_table$Date))
```

```
## # A tibble: 52 x 2
##    Date       EPS
##    <date>     <chr>
##  1 2009-03-31 $0.38
##  2 2009-06-30 $0.25
##  3 2009-09-30 $0.48
##  4 2009-12-31 $0.37
##  5 2010-03-31 $0.36
##  6 2010-06-30 $0.30
##  7 2010-09-30 $0.67
##  8 2010-12-31 $0.29
##  9 2011-03-31 $0.42
## 10 2011-06-30 $0.33
## # ... with 42 more rows
```

#In order to merge this to the original data, we shoudl have all the dates in the year to be present or else we will be loosing lot of data. So I created a data frame with only dates and merged it with the web scrapped data.

```
Date<-seq(as.Date("2009-03-06"),as.Date("2022-03-31"),by = 1)
Dates_df<-as.data.frame(Date)
```

#2.2 Handling Missing Values in Web Scrapped data

#I used LOCF method to handle the missing values as EPS for the next quarter is the same.

```
webscrap_CVS_table_merged <- merge(x = webscrap_CVS_table,y = Dates_df,  by = 'Date', all.y = TRUE)
webscrap_CVS_table_merged <- webscrap_CVS_table_merged %>%
  fill(Date, EPS, .direction = "up")
webscrap_CVS_table_merged<-webscrap_CVS_table_merged%>%mutate(EPS = EPS %>% str_remove_all("\\$"))
webscrap_CVS_table_merged<-webscrap_CVS_table_merged%>%mutate(EPS = as.numeric(EPS))
webscrap_CocaCola_table_merged <- merge(x = webscrap_CocaCola_table,y = Dates_df,  by = 'Date', all.y =
webscrap_CocaCola_table_merged <- webscrap_CocaCola_table_merged %>%
  fill(Date, EPS, .direction = "up")
webscrap_CocaCola_table_merged<-webscrap_CocaCola_table_merged%>%mutate(EPS = EPS %>% str_remove_all("\
webscrap_CocaCola_table_merged<-webscrap_CocaCola_table_merged%>%mutate(EPS = as.numeric(EPS))
webscrap_Apple_table_merged <- merge(x = webscrap_Apple_table,y = Dates_df,  by = 'Date', all.y = TRUE)
webscrap_Apple_table_merged <- webscrap_Apple_table_merged %>%
  fill(Date, EPS, .direction = "up")
webscrap_Apple_table_merged<-webscrap_Apple_table_merged%>%mutate(EPS = EPS %>% str_remove_all("\\$"))
webscrap_Apple_table_merged<-webscrap_Apple_table_merged%>%mutate(EPS = as.numeric(EPS))
webscrap_Tesla_table_merged <- merge(x = webscrap_Tesla_table,y = Dates_df,  by = 'Date', all.y = TRUE)
webscrap_Tesla_table_merged <- webscrap_Tesla_table_merged %>%
  fill(Date, EPS, .direction = "up")
webscrap_Tesla_table_merged<-webscrap_Tesla_table_merged%>%mutate(EPS = EPS %>% str_remove_all("\\$"))
webscrap_Tesla_table_merged<-webscrap_Tesla_table_merged%>%mutate(EPS = as.numeric(EPS))
webscrap_Phillips66_table_merged <- merge(x = webscrap_Phillips66_table,y = Dates_df,  by = 'Date', all
webscrap_Phillips66_table_merged <- webscrap_Phillips66_table_merged %>%
  fill(Date, EPS, .direction = "up")
webscrap_Phillips66_table_merged<-webscrap_Phillips66_table_merged%>%mutate(EPS = EPS %>% str_remove_al
```

```
webscrap_Phillips66_table_merged<-webscrap_Phillips66_table_merged%>%mutate(EPS = as.numeric(EPS))
webscrap_Xcel_table_merged <- merge(x = webscrap_Xcel_table,y = Dates_df,  by = 'Date', all.y = TRUE)
webscrap_Xcel_table_merged <- webscrap_Xcel_table_merged %>%
  fill(Date, EPS, .direction = "up")
webscrap_Xcel_table_merged<-webscrap_Xcel_table_merged%>%mutate(EPS = EPS %>% str_remove_all("\\$"))
webscrap_Xcel_table_merged<-webscrap_Xcel_table_merged%>%mutate(EPS = as.numeric(EPS))
```

##Merging the Web scrapped data with the original data and dividing train and test data sets.

```
F_TSLA <- merge(TSLA, webscrap_Tesla_table_merged
, by = "Date")
#F_TSLA<-F_TSLA%>%mutate(TSLA.PE = ifelse(TSLA.EPS<=0, 0, TSLA.Close/TSLA.EPS))
F_TSLA_train <-head(F_TSLA,round(0.65*nrow(F_TSLA)))
F_TSLA_test <- tail(F_TSLA,round(0.30*nrow(F_TSLA)))
F_CVS <- merge(CVS, webscrap_CVS_table_merged
, by = "Date")
#F_CVS<-F_CVS%>%mutate(CVS.PE = ifelse(CVS.EPS<=0, 0, CVS.Close/CVS.EPS))
F_CVS_train <-head(F_CVS,round(0.65*nrow(F_CVS)))
F_CVS_test <- tail(F_CVS,round(0.30*nrow(F_CVS)))
F_KO <- merge(KO, webscrap_CocaCola_table_merged
, by = "Date")
#F_KO<-F_KO%>%mutate(KO.PE = ifelse(KO.EPS<=0, 0, KO.Close/KO.EPS))
F_KO_train <-head(F_KO,round(0.65*nrow(F_KO)))
F_KO_test <- tail(F_KO,round(0.30*nrow(F_KO)))

F_XEL <- merge(XEL, webscrap_Xcel_table_merged
, by = "Date")
#F_XEL<-F_XEL%>%mutate(XEL.PE = ifelse(XEL.EPS<=0, 0, XEL.Close/XEL.EPS))
F_XEL_train <-head(F_XEL,round(0.65*nrow(F_XEL)))
F_XEL_test <- tail(F_XEL,round(0.30*nrow(F_XEL)))
F_AAPL <- merge(AAPL, webscrap_Apple_table_merged
, by = "Date")
#F_AAPL<-F_AAPL%>%mutate(AAPL.PE = ifelse(AAPL.EPS<=0, 0, AAPL.Close/AAPL.EPS))
F_AAPL_train <-head(F_AAPL,round(0.65*nrow(F_AAPL)))
F_AAPL_test <- tail(F_AAPL,round(0.30*nrow(F_AAPL)))
F_PSX <- merge(PSX, webscrap_Phillips66_table_merged
, by = "Date")
#F_PSX<-F_PSX%>%mutate(PSX.PE = ifelse(PSX.EPS<=0, 0, PSX.Close/PSX.EPS))
F_PSX_train <-head(F_PSX,round(0.65*nrow(F_PSX)))
F_PSX_test <- tail(F_PSX,round(0.30*nrow(F_PSX)))
```

Combining all the companies data into a single data set which will make the EDA much easier.

```
F_df <- rbind(F_XEL,F_CVS,F_KO,F_AAPL,F_PSX,F_TSLA)
F_df_train <- rbind(F_XEL_train,F_CVS_train,F_KO_train,F_TSLA_train,F_AAPL_train,F_PSX_train)
F_df_test <- rbind(F_XEL_test,F_CVS_test,F_KO_test,F_TSLA_train,F_AAPL_test,F_PSX_test)
```

It is always easy to model numeric columns so giving ID to the Comapny_Names.

```
F_df <- transform(F_df,Company_ID = as.numeric(factor(Company_Name)))
head(F_df,2)
```

```
##         Date  Open  High  Low Close  Volume Adj.Close Company_Name Growth
## 1 2009-03-06 16.61 17.01 16.2 16.57 4651400  10.30777          XEL      0
## 2 2009-03-07    NA    NA   NA    NA      NA        NA          XEL     NA
##   TrueRange  EPS Company_ID
## 1  0.809999 0.38          6
## 2        NA 0.38          6
```

```
F_df_train <- transform(F_df_train,Company_ID = as.numeric(factor(Company_Name)))

F_df_test <- transform(F_df_test,Company_ID = as.numeric(factor(Company_Name)))
```

#2.3 DATA CLEANING

```
sum(is.na(as.matrix(F_df)))
```

```
## [1] 67538
```

Using Spline interpolation: This method rely on the assumption that adjacent observations are similar to one another.

In time series data, if there are missing values, there are two ways to deal with the incomplete data:

#1. omit the entire record that contains information. #2. Impute the missing information. ### Since the time series data has temporal property, only some of the statistical methodologies are appropriate for time series data.

```
require(zoo)
#df <- df%>%mutate(rolling_7days = zoo::rollmean(Close, k = 7, fill = NA))
df1 <- F_df %>%group_by(Company_Name)%>%mutate(Close = na.spline(Close),Open = na.spline(Open),High = na
df1 <- df1%>%mutate(rolling_7days = zoo::rollmean(Close, k = 7, fill = NA))
df1_train <- F_df_train %>%group_by(Company_Name)%>%mutate(Close = na.spline(Close),Open = na.spline(Op
df1_train <- df1%>%mutate(rolling_7days = zoo::rollmean(Close, k = 7, fill = NA))
df1_test <- F_df_test %>%group_by(Company_Name)%>%mutate(Close = na.spline(Close),Open = na.spline(Open
df1_test <- df1_test%>%mutate(rolling_7days = zoo::rollmean(Close, k = 7, fill = NA))
head(df1,4)
```

```
## # A tibble: 4 x 13
##   Date        Open  High   Low Close   Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>     <dbl> <chr>         <dbl>
## 1 2009-03-06  16.6  17.0  16.2  16.6 4651400      10.3  XEL              0
## 2 2009-03-07  16.1  16.7  15.6  15.7 1088589.      9.77 XEL             NA
## 3 2009-03-08  16.1  16.5  15.7  15.7 1148269.      9.78 XEL             NA
## 4 2009-03-09  16.4  16.5  16.1  16.2 2951500      10.1  XEL              0
## # ... with 4 more variables: TrueRange <dbl>, EPS <dbl>, Company_ID <dbl>,
## #   rolling_7days <dbl>
```

```
df1 <- df1%>%group_by(Company_Name)%>%mutate(rolling_7days = na.spline(rolling_7days))%>%ungroup()

df1_train <- df1_train%>%group_by(Company_Name)%>%mutate(rolling_7days = na.spline(rolling_7days))%>%un

df1_test <- df1_test%>%group_by(Company_Name)%>%mutate(rolling_7days = na.spline(rolling_7days))%>%ungr
head(df1)
```

```
## # A tibble: 6 x 13
##   Date        Open  High   Low Close   Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>     <dbl> <chr>         <dbl>
## 1 2009-03-06  16.6  17.0  16.2  16.6 4651400      10.3  XEL               0
## 2 2009-03-07  16.1  16.7  15.6  15.7 1088589.      9.77 XEL              NA
## 3 2009-03-08  16.1  16.5  15.7  15.7 1148269.      9.78 XEL              NA
## 4 2009-03-09  16.4  16.5  16.1  16.2 2951500      10.1  XEL               0
## 5 2009-03-10  16.5  16.6  16.3  16.6 4274400      10.3  XEL               1
## 6 2009-03-11  16.0  16.9  16.0  16.6 3410500      10.4  XEL               1
## # ... with 4 more variables: TrueRange <dbl>, EPS <dbl>, Company_ID <dbl>,
## #   rolling_7days <dbl>
```

For the growth column, as it is based on whether or not grown, I have replaced NA values with 0, where the corresponding columns had NA values.

```
df1 <- df1 %>% mutate_at(9, ~replace_na(.,0))
df1_train <- df1_train %>% mutate_at(9, ~replace_na(.,0))
df1_test <- df1_test %>% mutate_at(9, ~replace_na(.,0))
head(df1)
```

```
## # A tibble: 6 x 13
##   Date        Open  High   Low Close   Volume Adj.Close Company_Name Growth
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>     <dbl> <chr>         <dbl>
## 1 2009-03-06  16.6  17.0  16.2  16.6 4651400      10.3  XEL               0
## 2 2009-03-07  16.1  16.7  15.6  15.7 1088589.      9.77 XEL               0
## 3 2009-03-08  16.1  16.5  15.7  15.7 1148269.      9.78 XEL               0
## 4 2009-03-09  16.4  16.5  16.1  16.2 2951500      10.1  XEL               0
## 5 2009-03-10  16.5  16.6  16.3  16.6 4274400      10.3  XEL               1
## 6 2009-03-11  16.0  16.9  16.0  16.6 3410500      10.4  XEL               1
## # ... with 4 more variables: TrueRange <dbl>, EPS <dbl>, Company_ID <dbl>,
## #   rolling_7days <dbl>
```

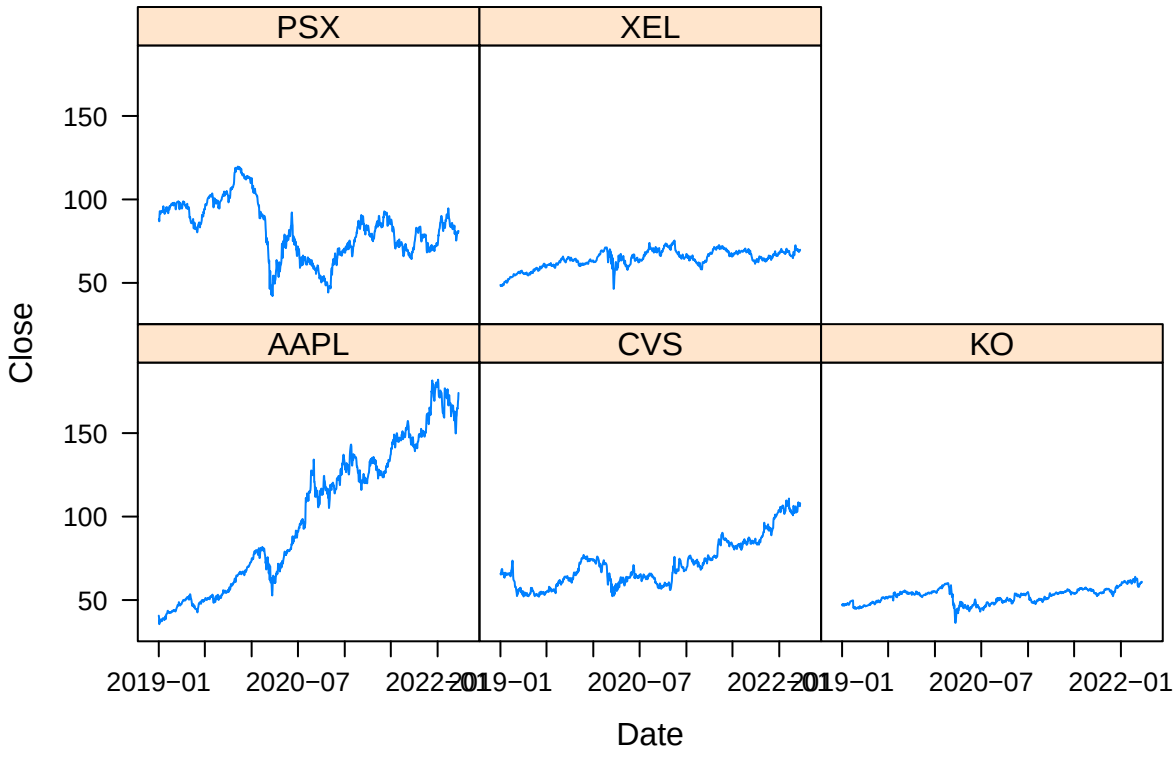
#3. EXPLORATORY DATA ANALYSIS:

I dropped out two companies(GOOG, TSLA) because they both had very high Closing price which made the visualization uninterpretable.

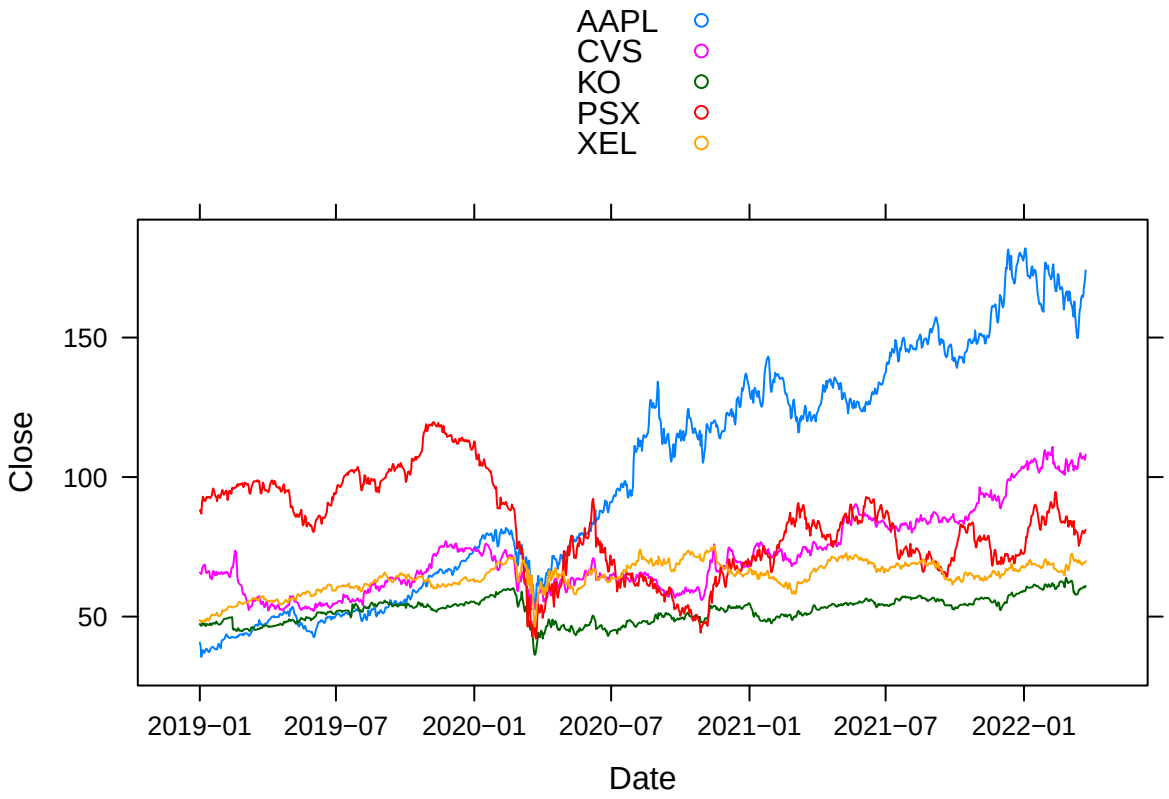
```
set.seed(1)
df2 <- df1%>%group_by(Date,Company_ID)%>%summarise(EPS,Company_Name,Date,Close)%>%filter(Date>='2019-01
```

```
## `summarise()` has grouped output by 'Date'. You can override using the
## `.groups` argument.
```

```
xyplot(x=Close ~ Date|df2$Company_Name, type='l', data = df2, scales=list(alternating=FALSE, tck=1:0))
```



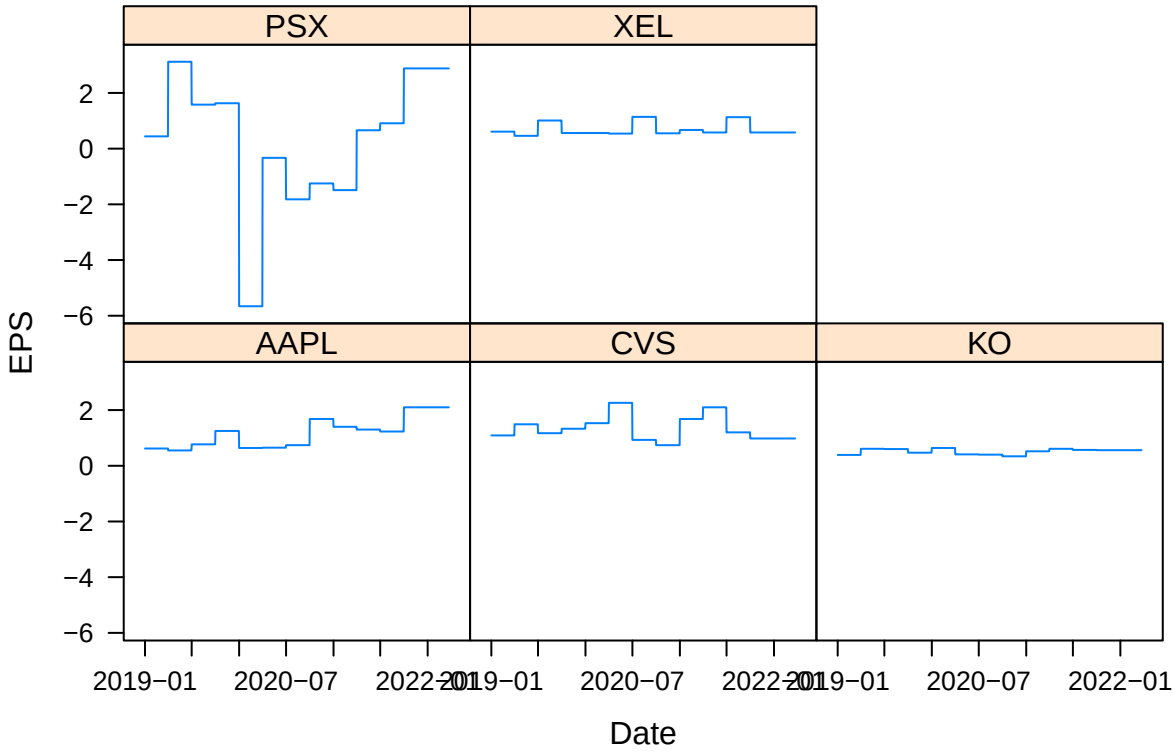

```
xyplot(Close ~ Date, groups=df2$Company_Name, data = df2, type = "l",group = df2$Company_Name, auto.key
```



Above, Graph represents Close price with respect to Dates for 5 companies of different sectors.

Another plot to represent EPS wrt. Date

```
xyplot(EPS ~  Date|df2$Company_Name, data=df2,  type='l',scales=list(alternating=FALSE, tck=1:0))
```

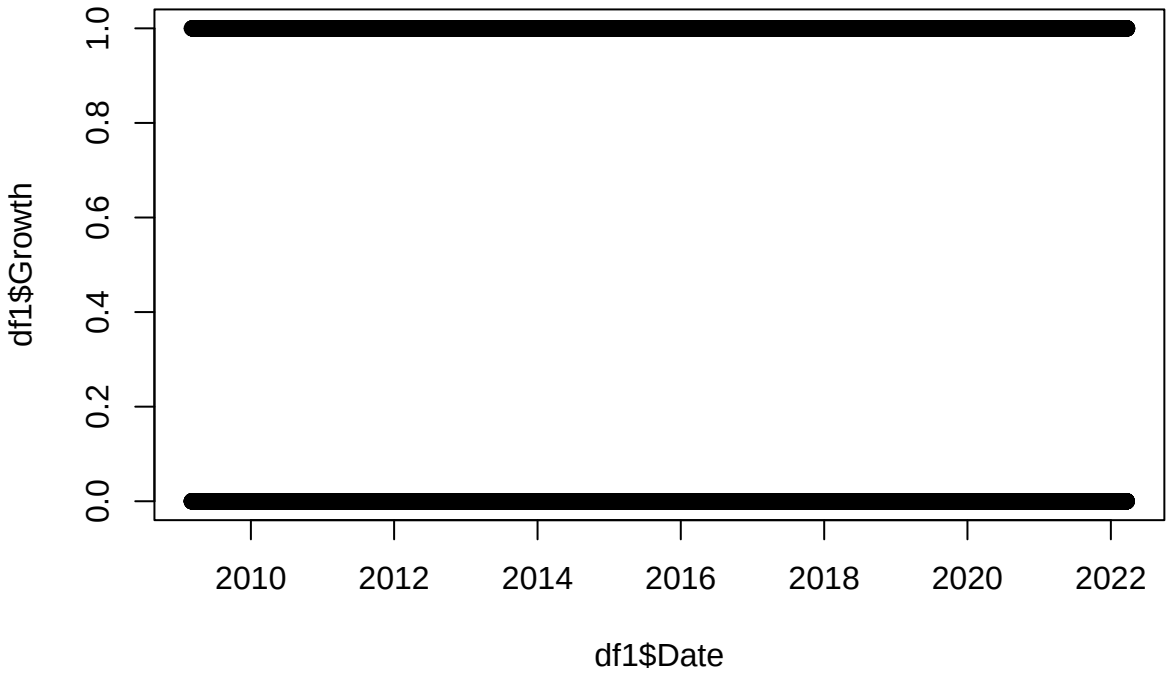


The above plot represents the EPS with respect to the Date and how it varies for different sector. We can visibly see that PSX(Phillips66-energy sector), has hiked after Feb 2022(after the war outbreak)

Plotting Growth with respect to the date.

Growth is a binomial(0 or 1) attribute.

```
plot(df1$Date,df1$Growth)
```



TIME SERIES FORECASTING:

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

#Here for this forecasting I have taken three companies each of differnt sector that will clearly indicate the difference in the Stock Closing price

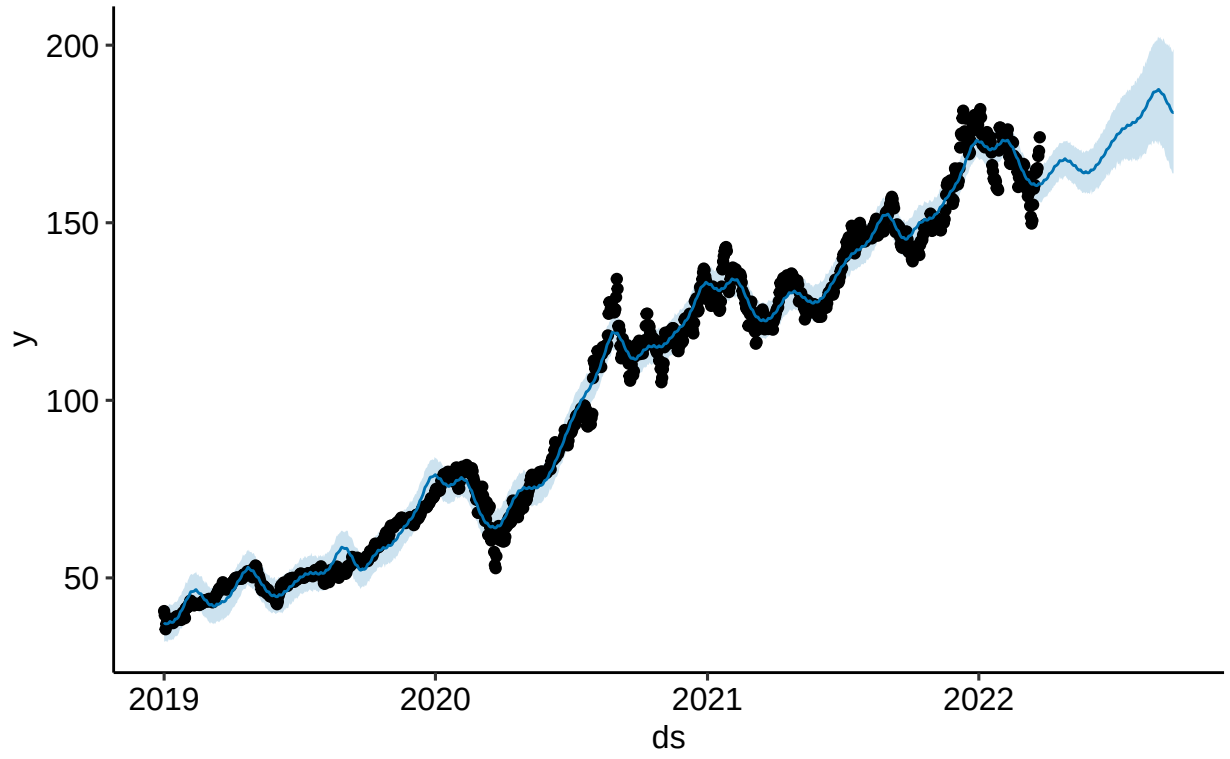
```
ds <- df2$Date
ID <- df2$Company_ID
y <- df2$Close
df_prophet <- data.frame(ds, ID, y)
df_prophet_CVS <- df_prophet%>%filter(ID==1)
df_prophet_PSX <- df_prophet%>%filter(ID==3)
df_prophet_XEL <- df_prophet%>%filter(ID==4)
head(df_prophet_XEL)
```

```
##           ds ID        y
## 1 2019-01-01  4 87.95549
## 2 2019-01-02  4 88.12000
## 3 2019-01-03  4 86.90000
## 4 2019-01-04  4 90.84000
## 5 2019-01-05  4 92.95106
## 6 2019-01-06  4 92.49105
```

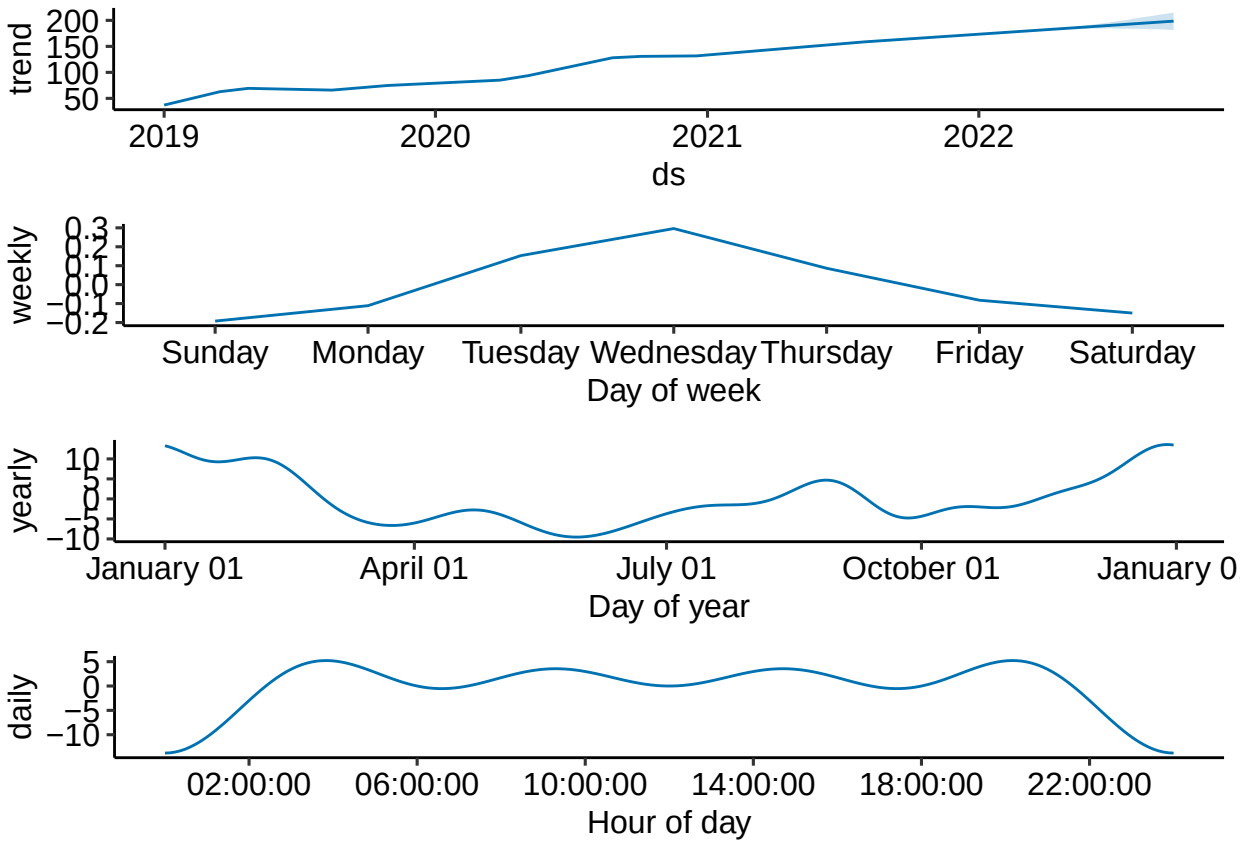
We need to first make a Prophet model (m) based on the data and have it make an empty future dataframe (future) for our desired number of periods (days in our case) that will be forecast. Here I have taken 90days i.e 3 months.

```
m_CVS <- prophet(df_prophet_CVS, daily.seasonality=TRUE)
m_PSX <- prophet(df_prophet_PSX, daily.seasonality=TRUE)
m_XEL <- prophet(df_prophet_XEL, daily.seasonality=TRUE)

future_CVS <- make_future_dataframe(m_CVS, periods=180)
forecast_CVS <- predict(m_CVS, periods=180, future_CVS)
plot(m_CVS, forecast_CVS)
```



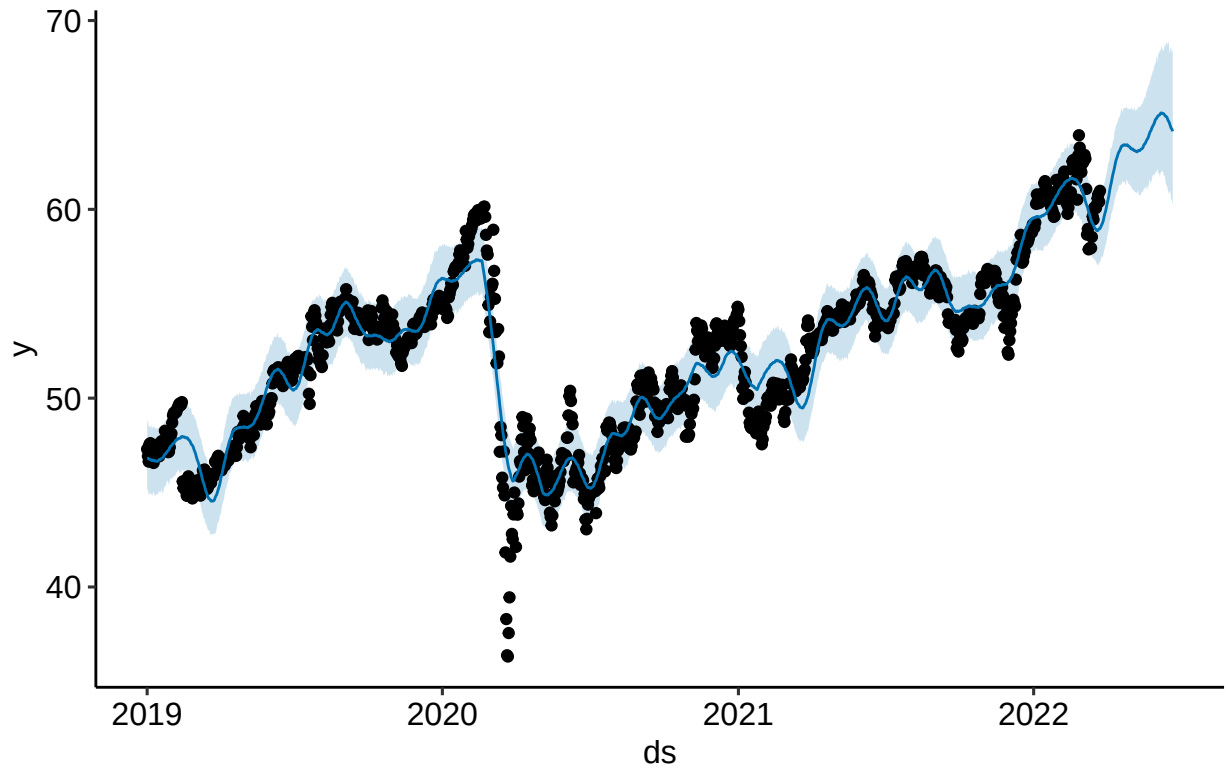

```
prophet_plot_components(m_CVS, forecast_CVS)
```



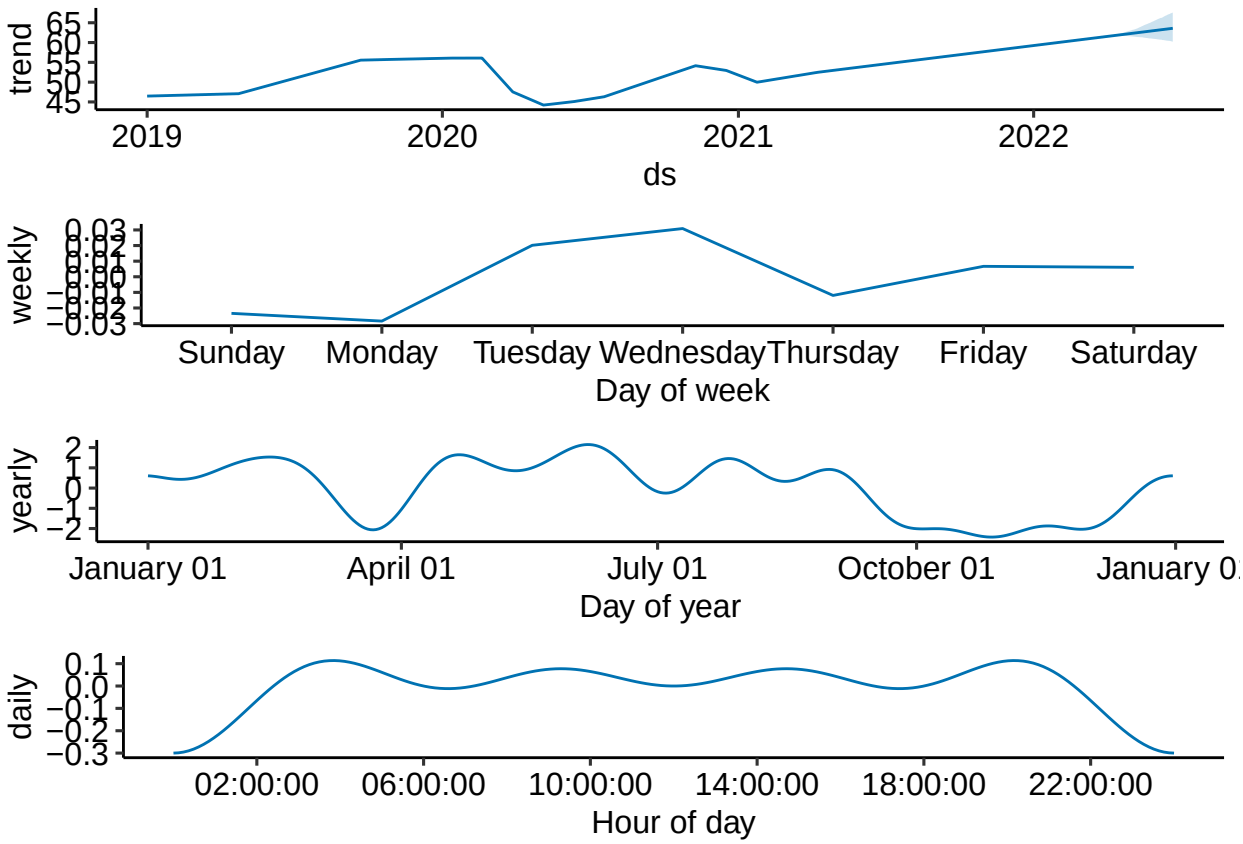
The above plot is the Forecasting plot for CVS pharmacy(Health care Sector). We can see that there was an unexpected price drop in Feb 2022 by looking at the black dots dropping below the forecast (blue cast window).

We need to first make a Prophet model (m) based on the data and have it make an empty future dataframe (future) for our desired number of periods (days in our case) that will be forecast. Here I have taken 180days i.e 6 months.

```
future_PSX <- make_future_dataframe(m_PSX, periods=90)
forecast_PSX <- predict(m_PSX, periods=90, future_PSX)
plot(m_PSX, forecast_PSX)
```

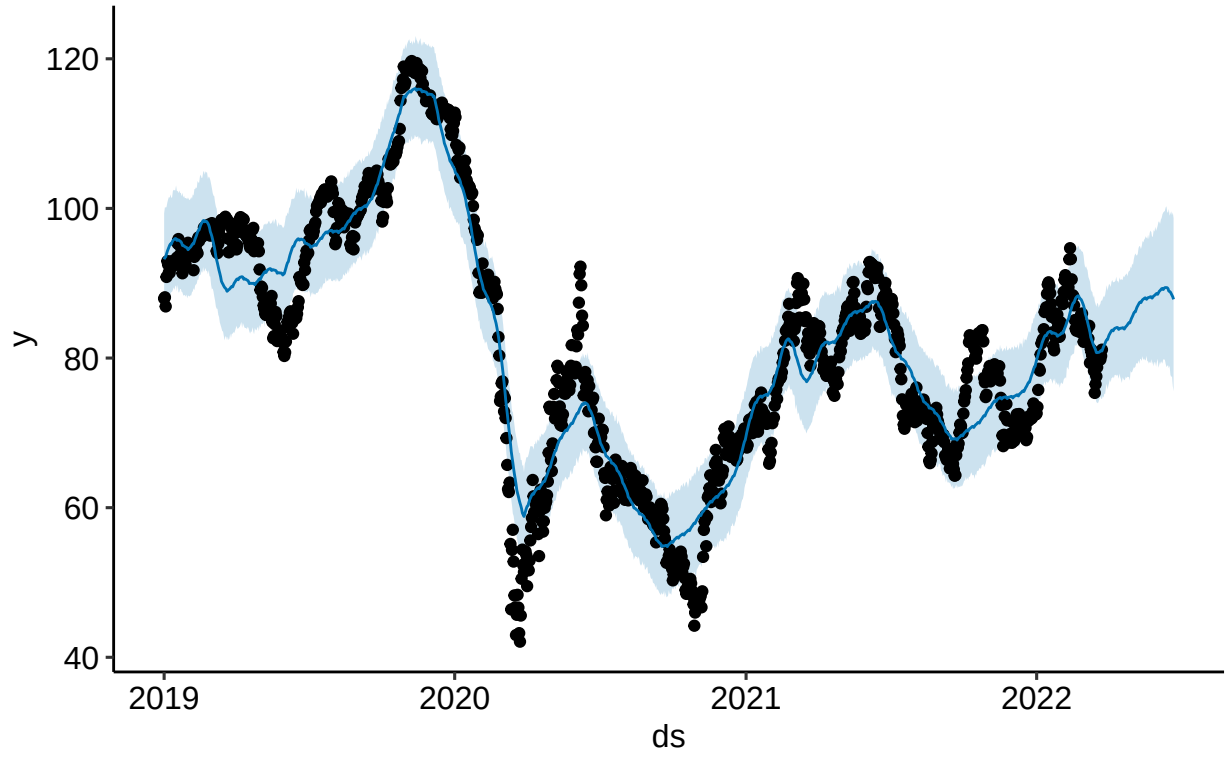



```
prophet_plot_components(m_PSX, forecast_PSX)
```

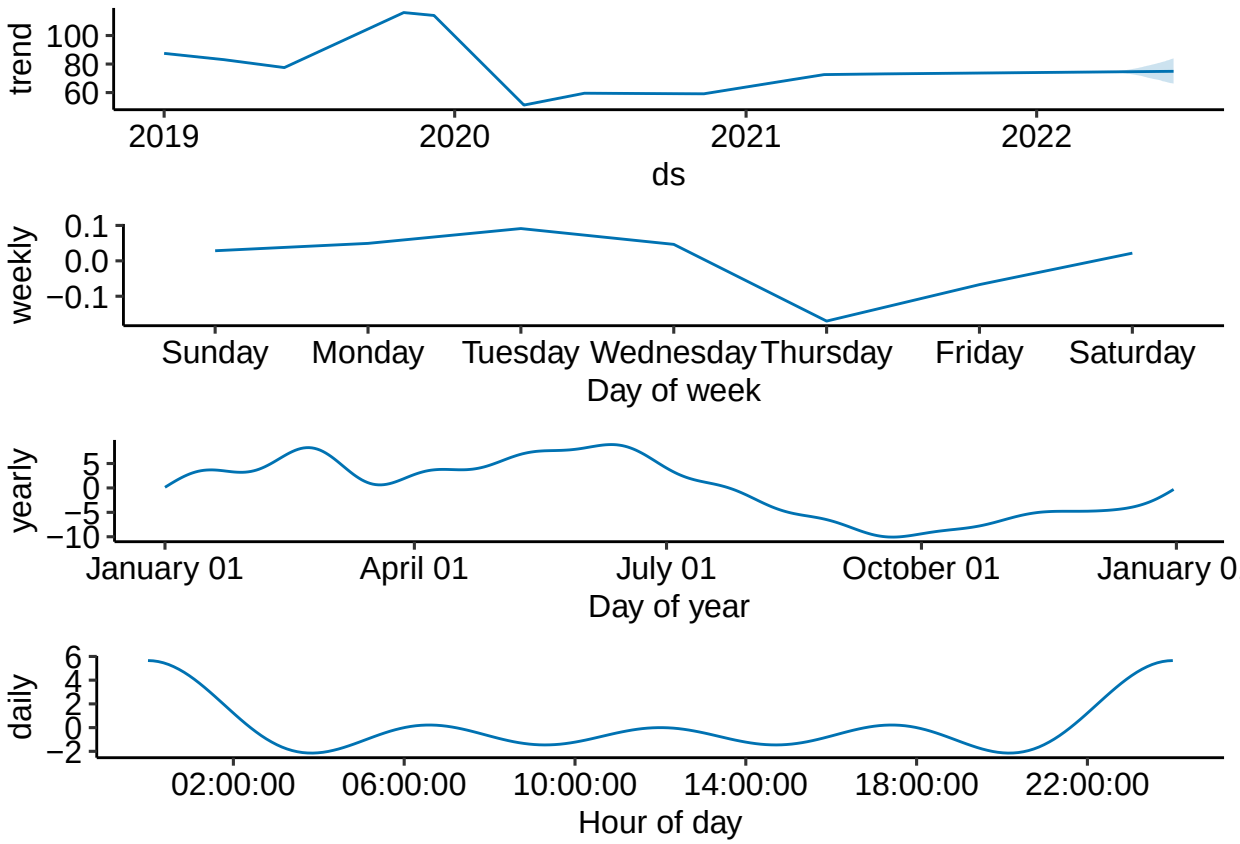


The above plot is the Forecasting plot for Phillips66 or PSX (Energy Sector). We can see that there was a price hike in Feb 2021 by looking at the black dots hiking below the forecast (blue cast window).

```
future_XEL <- make_future_dataframe(m_XEL, periods=90)
forecast_XEL <- predict(m_XEL, periods=90, future_XEL)
plot(m_XEL, forecast_XEL)
```




```
prophet_plot_components(m_XEL, forecast_XEL)
```



The above plot is the Forecasting plot for Xcel Energy or XEL or PSX (utilities Sector). We can see that there was price drop in Feb 2021 by looking at the black dots dropping below the forecast (blue cast window).

In the above Forecasting plots, 1. the black dots represent actual measurements 2. blue line displays Prophet's forecast 3. light blue window indicates uncertainty intervals

Prophet time series for grouped data i.e representing all the trends in a single data. However I wasn't able to find a solution for plotting the grouped data forecasting.

```
data = df_prophet %>%
       group_by(ID) %>%
       do(predict(prophet(.), make_future_dataframe(prophet(.)

                                                    , periods = 7)))
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
data <- data %>% group_by(ID) %>%
         top_n(7, ds)
tail(data[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
## # A tibble: 6 x 4
##   ds                   yhat yhat_lower yhat_upper
##   <dttm>              <dbl>      <dbl>      <dbl>
## 1 2022-03-26 00:00:00  66.6       64.2       69.3
## 2 2022-03-27 00:00:00  66.6       64.1       69.0
## 3 2022-03-28 00:00:00  66.5       64.0       69.1
## 4 2022-03-29 00:00:00  66.5       63.9       69.0
## 5 2022-03-30 00:00:00  66.6       64.3       69.1
## 6 2022-03-31 00:00:00  66.7       64.1       69.3
```

ds — date being forecast. yhat — prediction for y value (number of views) that day. yhat_lower — lowest expected value for that day's predicted y value range. yhat_upper — highest expected value for that day's predicted y value range.

#4. CANDLESTICK CHART

For a detailed understanding of the Close, Open, High, Low and Date of the stock market data.

I have visualized Phillips66(PSX) and Xcel Energy(XEL) using candelstick cahrt.

```
df1_candlestick_PSX <- df1%>%filter(Company_Name == 'PSX'&Date>='2022-01-01')
fig1 <- df1_candlestick_PSX %>%plot_ly(x = ~Date, type = "candlestick",
                        open = ~df1_candlestick_PSX$Open,
                        close = ~df1_candlestick_PSX$Close,
                        high = ~df1_candlestick_PSX$High,
                        low = ~df1_candlestick_PSX$Low)
fig1 <- fig1%>% layout(title = "Basic Candlestick Chart Phillips66")
fig1
```

```
df1_candlestick_XEL <- df1%>%filter(Company_Name ==
                                      'XEL'&Date>='2022-01-01')
fig2 <- df1_candlestick_XEL %>%plot_ly(x = ~Date, type = "candlestick",
                        open = ~df1_candlestick_XEL$Open,
                        close = ~df1_candlestick_XEL$Close,
                        high = ~df1_candlestick_XEL$High,
                        low = ~df1_candlestick_XEL$Low)
fig2 <- fig2%>% layout(title = "Basic Candlestick Chart Xcel Energy")
fig2
```

#5. CORRELATION MATRIX

##5.1 Compute correlation matrix

A correlation matrix is a table of correlation coefficients for a set of variables used to determine if a relationship exists between the variables. The coefficient indicates both the strength of the relationship as well as the direction (positive vs. negative correlations). In this post I show you how to calculate and visualize a correlation matrix using R.

```
corelation <- df1%>%summarise(Growth, TrueRange,High, Low, Open, Volume ,
                              Close, Adj.Close, Company_ID,
                              rolling_7days, EPS)
str(corelation)
```

```
## tibble [26,989 x 11] (S3: tbl_df/tbl/data.frame)
##  $ Growth       : num [1:26989] 0 0 0 0 1 1 1 1 0 0 ...
##  $ TrueRange    : num [1:26989] 0.81 1.063 0.825 0.44 0.35 ...
##  $ High         : num [1:26989] 17 16.7 16.5 16.5 16.6 ...
##  $ Low          : num [1:26989] 16.2 15.6 15.7 16.1 16.3 ...
##  $ Open         : num [1:26989] 16.6 16.1 16.1 16.4 16.5 ...
##  $ Volume       : num [1:26989] 4651400 1088589 1148269 2951500 4274400 ...
##  $ Close        : num [1:26989] 16.6 15.7 15.7 16.2 16.6 ...
##  $ Adj.Close    : num [1:26989] 10.31 9.77 9.78 10.07 10.34 ...
##  $ Company_ID   : num [1:26989] 6 6 6 6 6 6 6 6 6 6 ...
##  $ rolling_7days: num [1:26989] 18 17 16.5 16.4 16.4 ...
##  $ EPS          : num [1:26989] 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 0.38 ...
```

```
cormat <- round(cor(corelation),9)
head(cormat,3)
```

```
##                 Growth    TrueRange         High          Low         Open
## Growth     1.000000000 -0.004612235 -0.001656266 -0.00147247 -0.004603179
## TrueRange -0.004612235  1.000000000  0.857422456  0.84186055  0.852428178
## High      -0.001656266  0.857422456  1.000000000  0.99956436  0.999682282
##                Volume       Close   Adj.Close  Company_ID rolling_7days
## Growth    -0.01048267 0.001311063 0.000892527 0.001718231  -0.001077327
## TrueRange -0.03076921 0.849845287 0.852602266 0.119681244   0.852017696
## High      -0.10408194 0.999632036 0.998614301 0.104804168   0.999591851
##                  EPS
## Growth    0.01124221
## TrueRange 0.09413370
## High      0.19458514
```

Above is the simple correlation matrix showing the correlations between pairs of variables.

The package reshape is required to melt the correlation matrix.

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.1.3
```

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

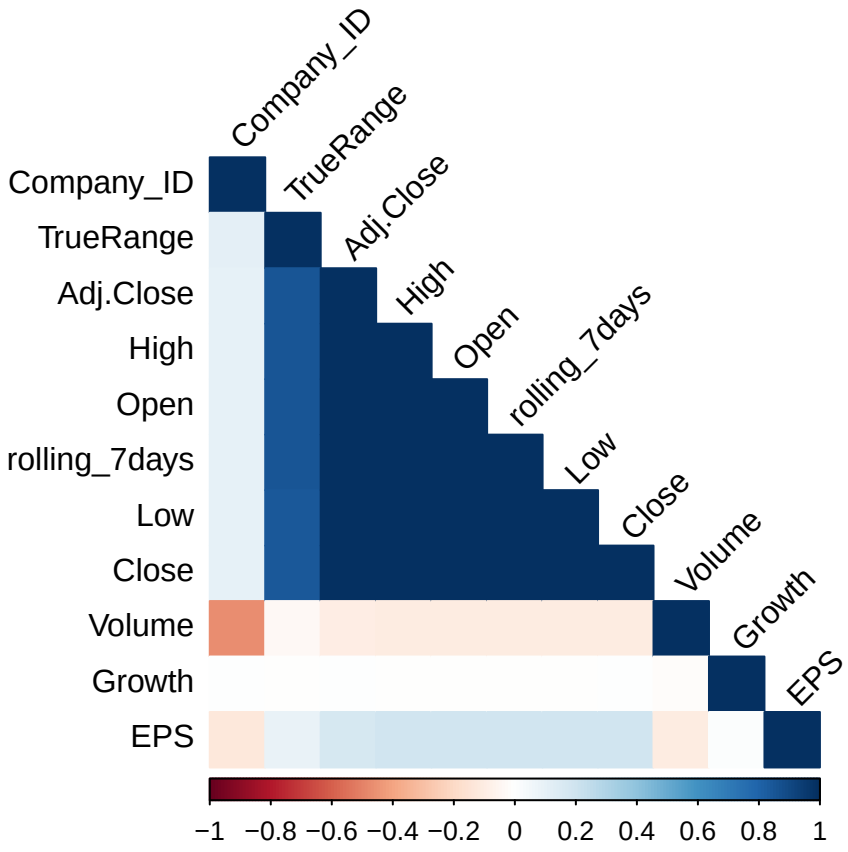
```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
melted_cormat <- melt(cormat)
head(melted_cormat)
```

```
##        Var1   Var2        value
## 1    Growth Growth  1.000000000
## 2 TrueRange Growth -0.004612235
## 3      High Growth -0.001656266
## 4       Low Growth -0.001472470
## 5      Open Growth -0.004603179
## 6    Volume Growth -0.010482674
```

##5.2 Visualizing the correlation matrix

```
corrplot(cormat, method="color",type = "lower", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



This relation can be expressed as a range of values expressed within the interval [-1, 1]. The value -1 indicates a perfect non-linear (negative) relationship, 1 is a perfect positive linear relationship and 0 is an intermediate between neither positive nor negative linear interdependency. However, a value of 0 doesn't indicate the variables to be independent of each other completely.

#6. MODELLING

##6.1 Linear Regression: #for EPS as the target and all the other variables as the predictors. Using OLS to visualize in table how the model will be after adding or removing any predictor.

```
library(olsrr)
```

```
## Warning: package 'olsrr' was built under R version 4.1.3
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:MASS':
##
##     cement
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(car)
F_df_PSX <- df1%>%filter(Company_Name=='PSX')

F_df_train_PSX <- df1_train%>%filter(Company_Name=='PSX')

F_df_test_PSX <- df1_test%>%filter(Company_Name=='PSX')

model_PSX <- lm(EPS ~ Volume+Date+Close+High, data = F_df_train_PSX)
ols_step_best_subset(model_PSX)
```

```
##        Best Subsets Regression
## -------------------------------------
## Model Index    Predictors
## -------------------------------------
##      1         Date
##      2         Date Close
##      3         Date Close High
##      4         Volume Date Close High
## -------------------------------------
##
##                                                    Subsets Regression Summary
## ---------------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square      C(p)         AIC          SBIC          SBC
## ---------------------------------------------------------------------------------------------------------
##   1        0.0584      0.0581      0.0574    615.1204    14584.3043    4270.8418    14602.8986    11
##   2        0.1943      0.1939      0.1929      4.3824    14019.8311    3706.9886    14044.6235    10
##   3        0.1950      0.1944      0.1926      3.1033    14018.5491    3705.7146    14049.5395    10
##   4        0.1951      0.1942      0.1922      5.0000    14020.4456    3707.6142    14057.6342    10
## ---------------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
```

```
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```
vif(model_PSX)
```

```
##     Volume       Date      Close       High
##   1.258988   1.318547 351.028376 350.537027
```

Usnig VIF function identifying if any predictor has correlation. In the above model I have selected only the Phillips66 (PSX) which hiked after the conflict for modelling.

##6.2 Mean Squared Error

Calculating mean squared error to find whether this model is accurate or not.

```
mse_PSX<- mean((F_df_test_PSX$EPS - predict(model_PSX,
                                            newdata = F_df_test_PSX,
                                            type="response"))^2)
mse_PSX
```

```
## [1] 5.665055
```

##6.3 GENERALIZED LINEAR MODELLING(GLM)

###The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value. Here Growth variable follows binomial distribution

##6.3.1 MODEL-1 FOR GROWTH: ###Models for indicating the growth(as the target) or the stock price in a day for the PSX_train data.

```
glmod_PSX_growth <- glm(Growth~Date+Close+High+Volume+EPS+Low,
                        data = F_df_train_PSX, family = 'binomial' )
summary(glmod_PSX_growth)
```

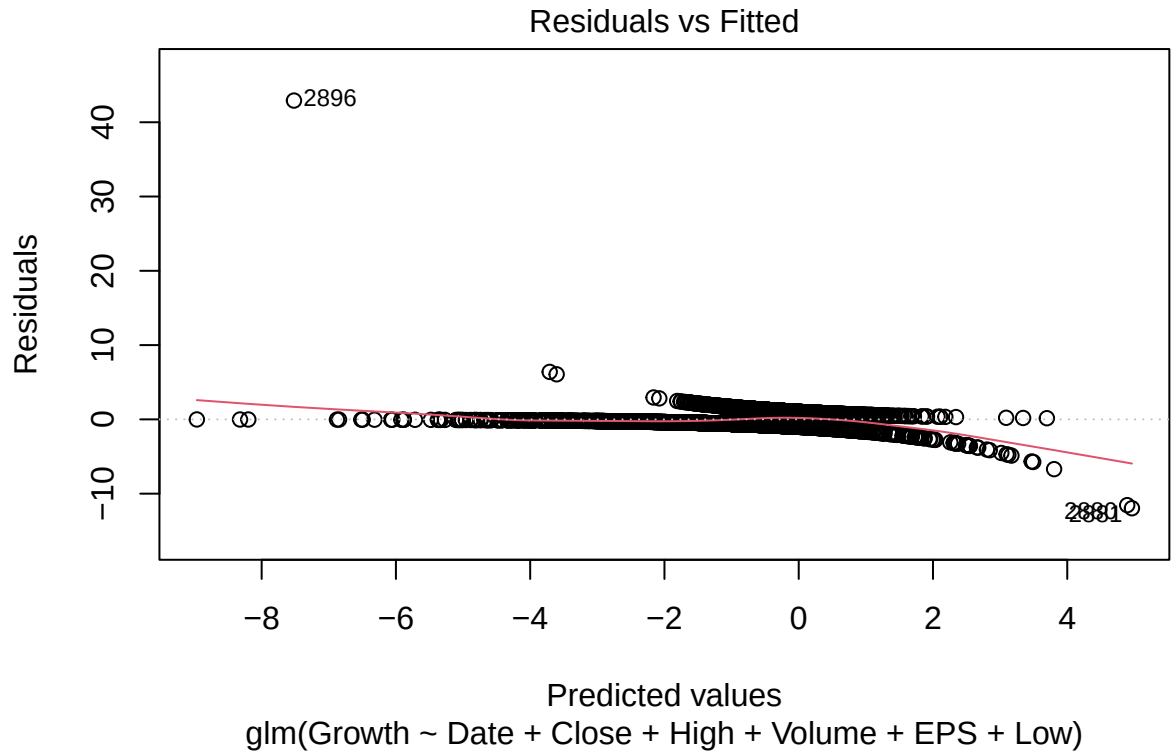
```
##
## Call:
## glm(formula = Growth ~ Date + Close + High + Volume + EPS + Low,
##     family = "binomial", data = F_df_train_PSX)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1530  -0.8372  -0.5207   1.0909   3.8779
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.325e-01  7.718e-01   0.431  0.66662
## Date        -4.856e-05  4.897e-05  -0.992  0.32136
## Close        1.540e+00  6.961e-02  22.116  < 2e-16 ***
## High        -8.586e-01  6.780e-02 -12.663  < 2e-16 ***
```

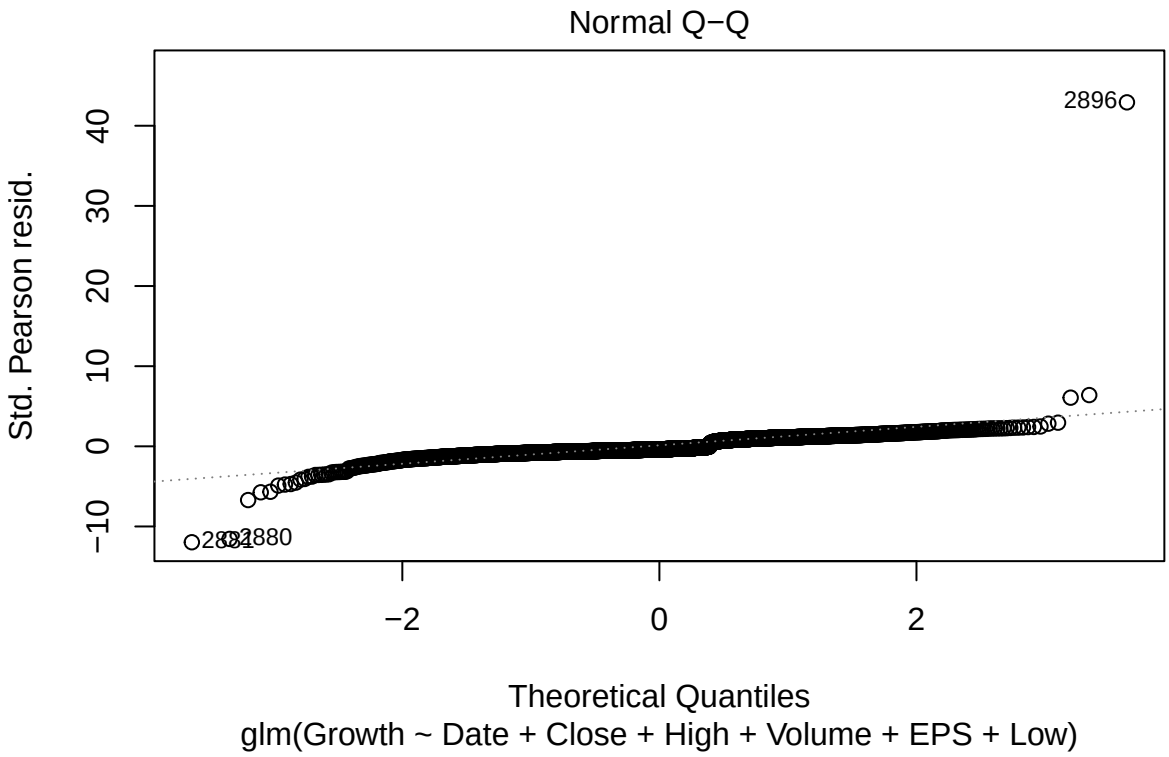
```
## Volume      -7.319e-08  3.039e-08  -2.408  0.01603 *
## EPS          7.575e-02  2.520e-02   3.007  0.00264 **
## Low         -6.812e-01  5.675e-02 -12.005  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4697.0  on 3633  degrees of freedom
## Residual deviance: 3952.9  on 3627  degrees of freedom
## AIC: 3966.9
##
## Number of Fisher Scoring iterations: 5
```

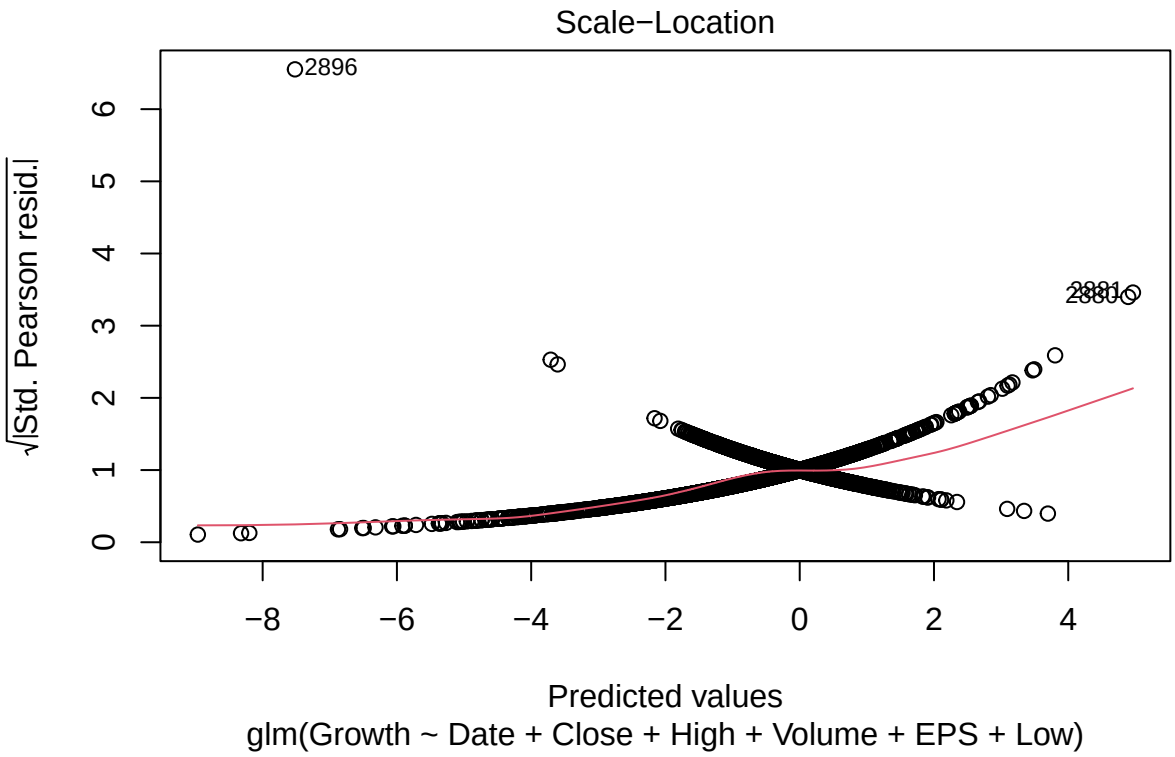
```
mse_glmod_PSX<- mean((F_df_test_PSX$Growth- predict(glmod_PSX_growth,
                                                    newdata = F_df_test_PSX,
                                                    type="response"))^2)
mse_glmod_PSX
```

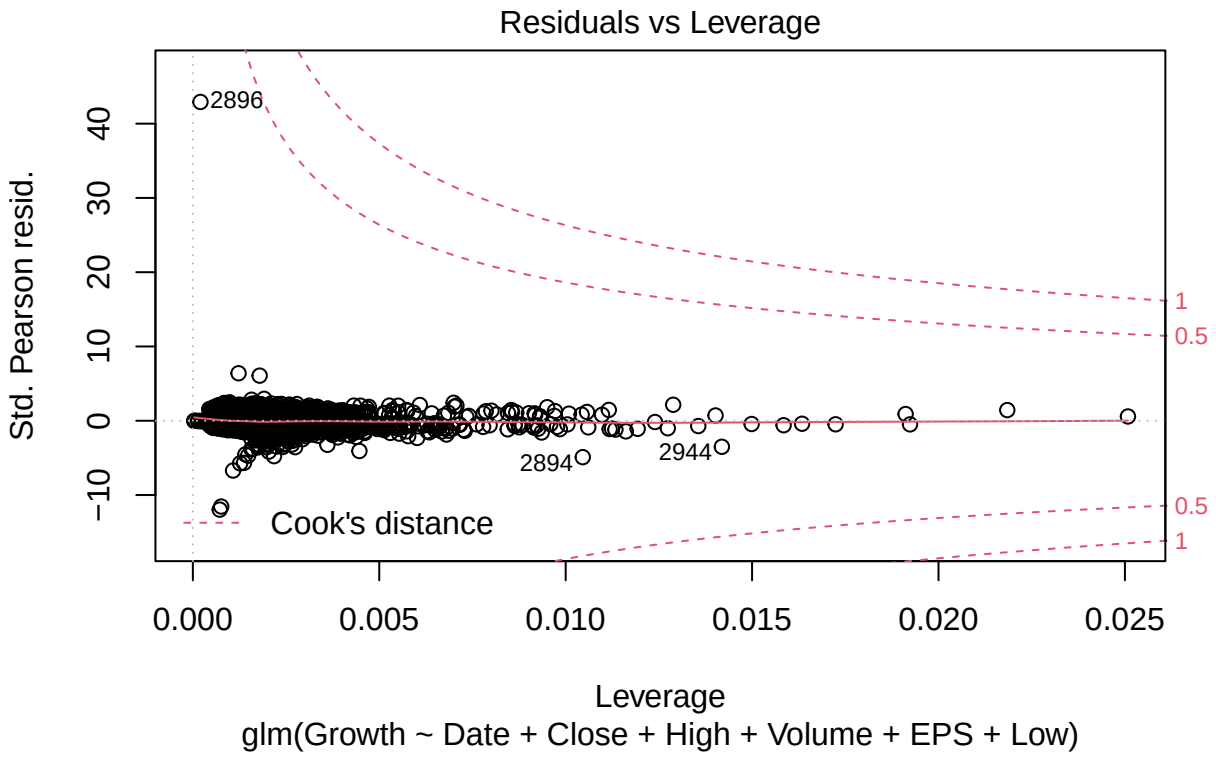
```
## [1] 0.1800426
```

```
plot(glmod_PSX_growth)
```




Normal Q−Q
Std. Pearson resid.
−10
0
10
20
30
40
2896
2880
−2
0
2
Theoretical Quantiles
glm(Growth ~ Date + Close + High + Volume + EPS + Low)


Scale−Location
2896
2881
2880
√|Std. Pearson resid.|
0
1
2
3
4
5
6
−8
−6
−4
−2
0
2
4
Predicted values
glm(Growth ~ Date + Close + High + Volume + EPS + Low)



From the above summary it is interpreted that Dated has no significant towards the target and the predictor High has little significant. The above value is the Mean Square error for the model which is .1800426

##6.3.2 MODEL-2 FOR GROWTH:

```
glmod_PSX1_growth <- glm(Growth~Close+High+Volume+Date+EPS,
                         data = F_df_train_PSX, family = 'binomial' )
summary(glmod_PSX1_growth)
```

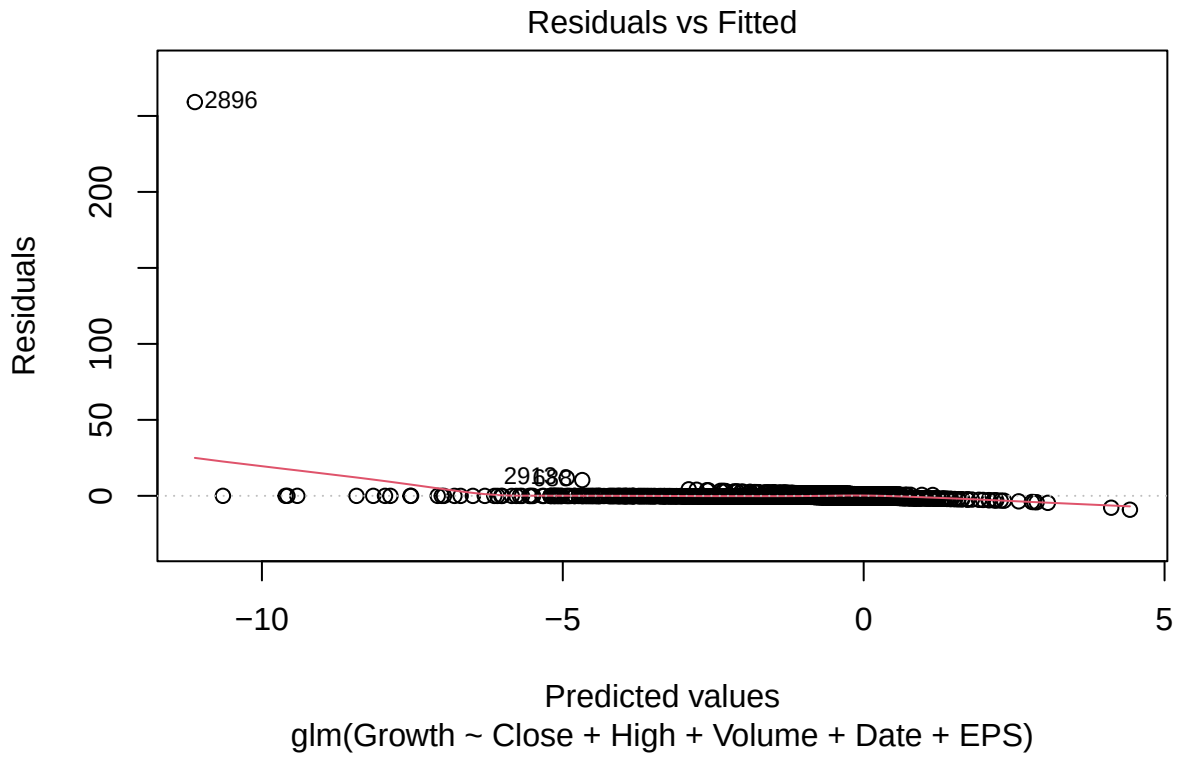
```
##
## Call:
## glm(formula = Growth ~ Close + High + Volume + Date + EPS, family = "binomial",
##     data = F_df_train_PSX)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9789  -0.9074  -0.5555   1.1599   4.7148
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.442e+00  7.168e-01  -3.406 0.000659 ***
## Close        1.240e+00  6.394e-02  19.391  < 2e-16 ***
## High        -1.237e+00  6.379e-02 -19.394  < 2e-16 ***
## Volume       8.912e-08  2.583e-08   3.451 0.000559 ***
## Date         1.277e-04  4.540e-05   2.812 0.004925 **
## EPS          4.448e-02  2.426e-02   1.834 0.066678 .
## ---
```

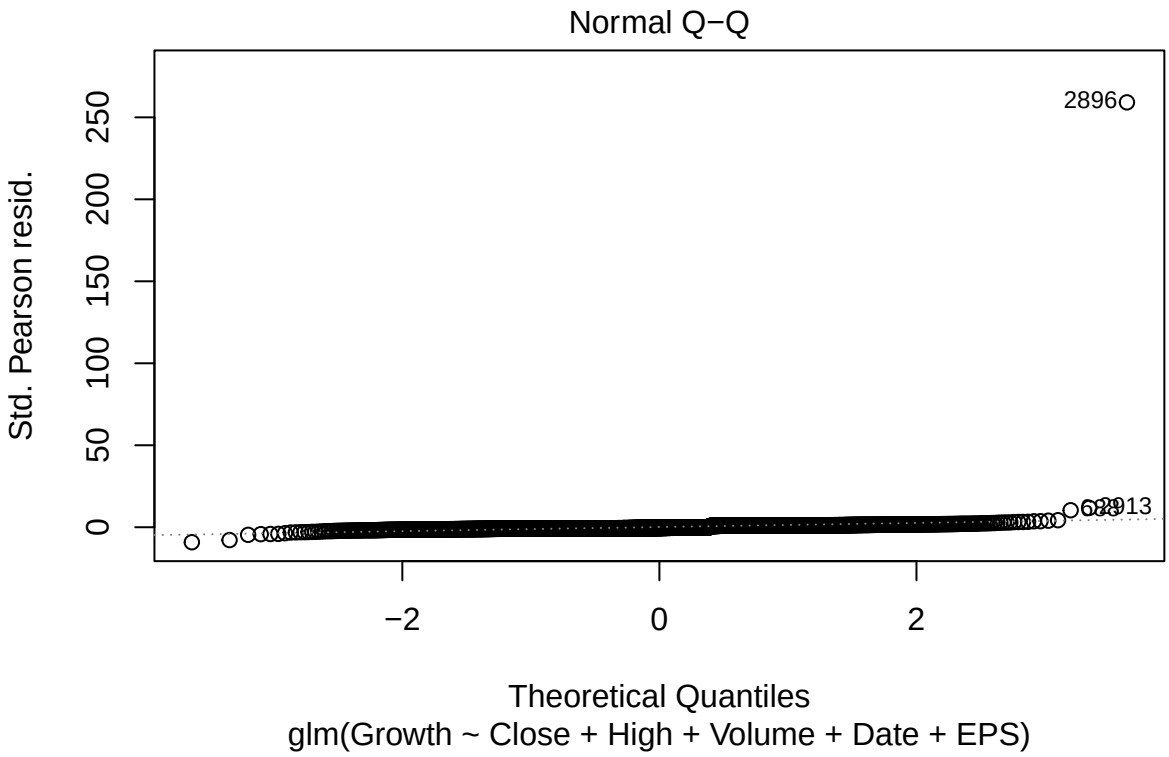
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4697.0  on 3633  degrees of freedom
## Residual deviance: 4115.9  on 3628  degrees of freedom
## AIC: 4127.9
##
## Number of Fisher Scoring iterations: 5
```

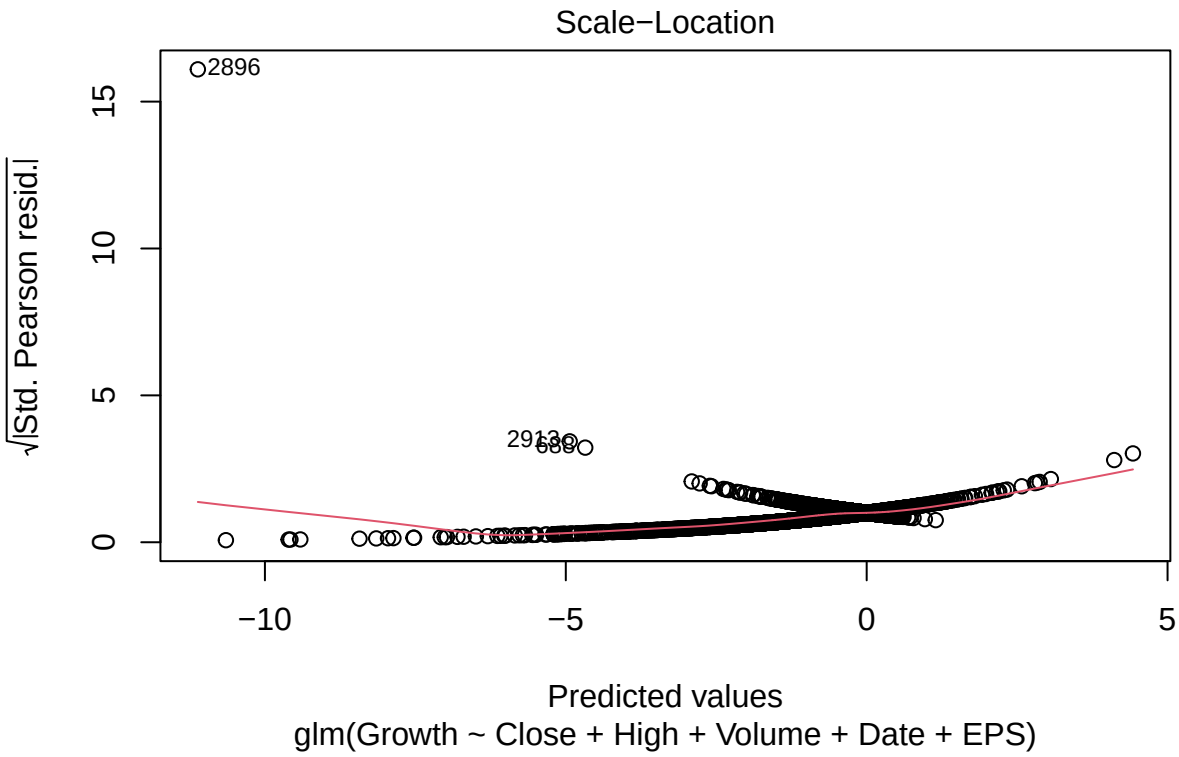
```
mse_glmod_PSX1<- mean((F_df_test_PSX$Growth- predict(glmod_PSX1_growth,
                                                     newdata = F_df_test_PSX,
                                                     type="response"))^2)
mse_glmod_PSX1
```

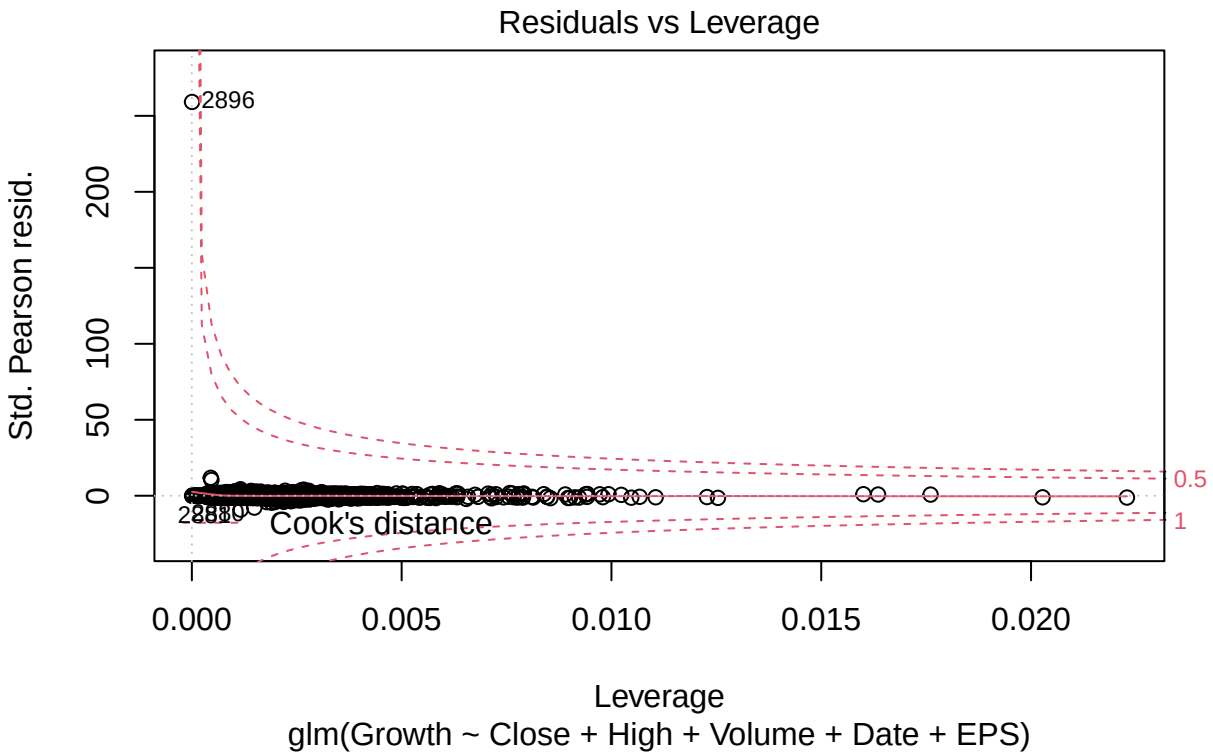
```
## [1] 0.193059
```

```
plot(glmod_PSX1_growth)
```




Normal Q−Q
Std. Pearson resid.
0
50
100
150
200
250
2896
629
2913
−2
0
2
Theoretical Quantiles
glm(Growth ~ Close + High + Volume + Date + EPS)


Scale−Location
2896
29139
√|Std. Pearson resid.|
0
5
10
15
−10
−5
0
5
Predicted values
glm(Growth ~ Close + High + Volume + Date + EPS)



The above value is the Mean Square error for the second model which is 0.193059.

##6.3.3 Comparing model using Anova ### The anova() function will take the model objects as arguments, and return an ANOVA testing whether the more complex model is significantly better at capturing the data than the simpler model.

```
anova(glmod_PSX_growth,glmod_PSX1_growth, test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: Growth ~ Date + Close + High + Volume + EPS + Low
## Model 2: Growth ~ Close + High + Volume + Date + EPS
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3627     3952.9
## 2      3628     4115.9 -1  -163.07 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on this anova Chisq test, smaller model is better than the bigger model.

##6.4 GLM FOR EPS AS TARGET

Taking EPS as the target variable and performing the glm modelling as we did for growth as the target

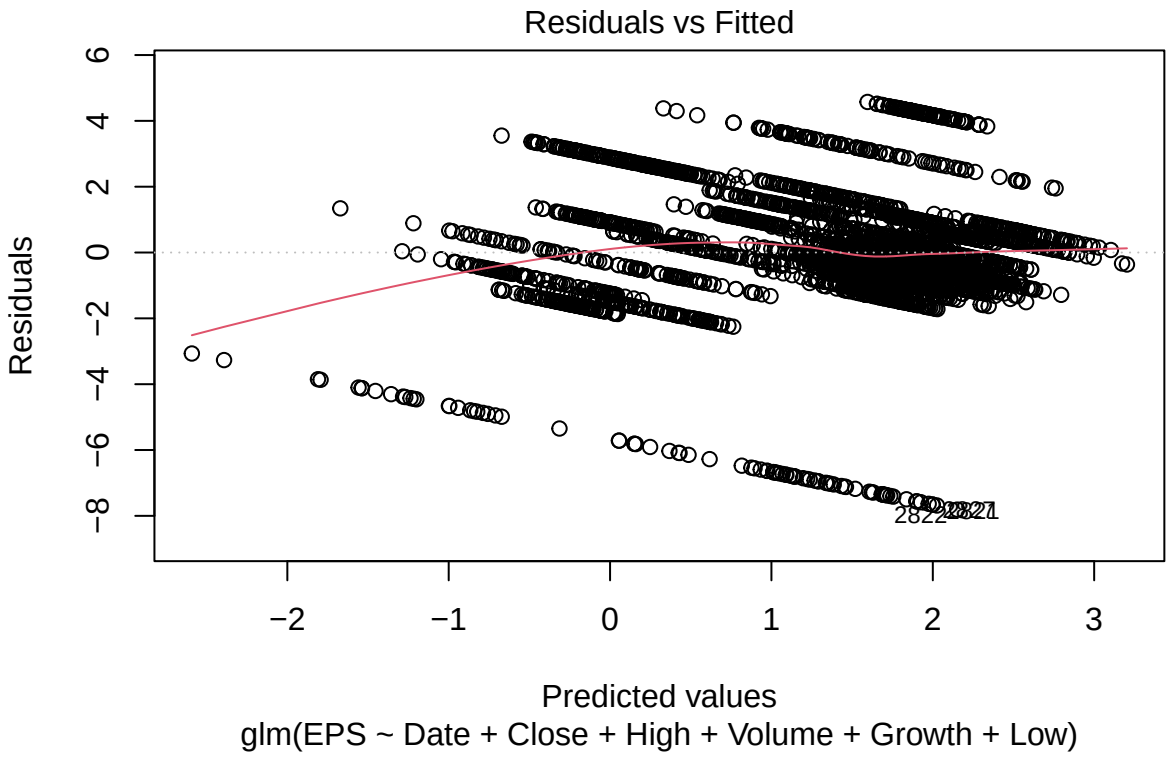
```
glmod_EPS_PSX <- glm(EPS~Date+Close+High+Volume+Growth+Low,
                 data = F_df_train_PSX, family = 'gaussian' )
summary(glmod_EPS_PSX)
```

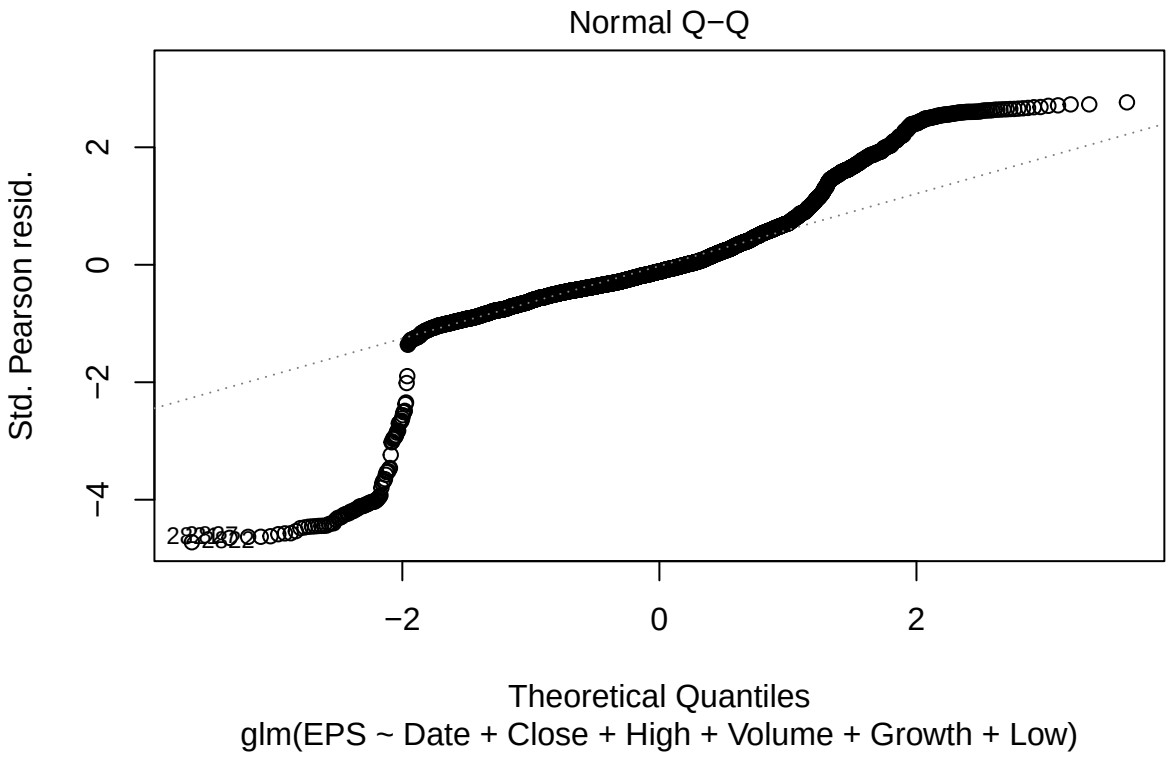
```
##
## Call:
## glm(formula = EPS ~ Date + Close + High + Volume + Growth + Low,
##     family = "gaussian", data = F_df_train_PSX)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.8112  -0.7238  -0.1923   0.6511   4.5743
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.999e+00  5.141e-01  19.449  < 2e-16 ***
## Date        -6.845e-04  3.173e-05 -21.573  < 2e-16 ***
## Close       -6.804e-02  3.784e-02  -1.798 0.072268 .
## High        -1.173e-01  3.261e-02  -3.597 0.000326 ***
## Volume       4.573e-08  2.033e-08   2.249 0.024574 *
## Growth       1.635e-01  6.321e-02   2.587 0.009709 **
## Low          2.274e-01  3.615e-02   6.290 3.55e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.73931)
##
##     Null deviance: 12481.9  on 3633  degrees of freedom
## Residual deviance:  9935.5  on 3627  degrees of freedom
## AIC: 13984
##
## Number of Fisher Scoring iterations: 2
```

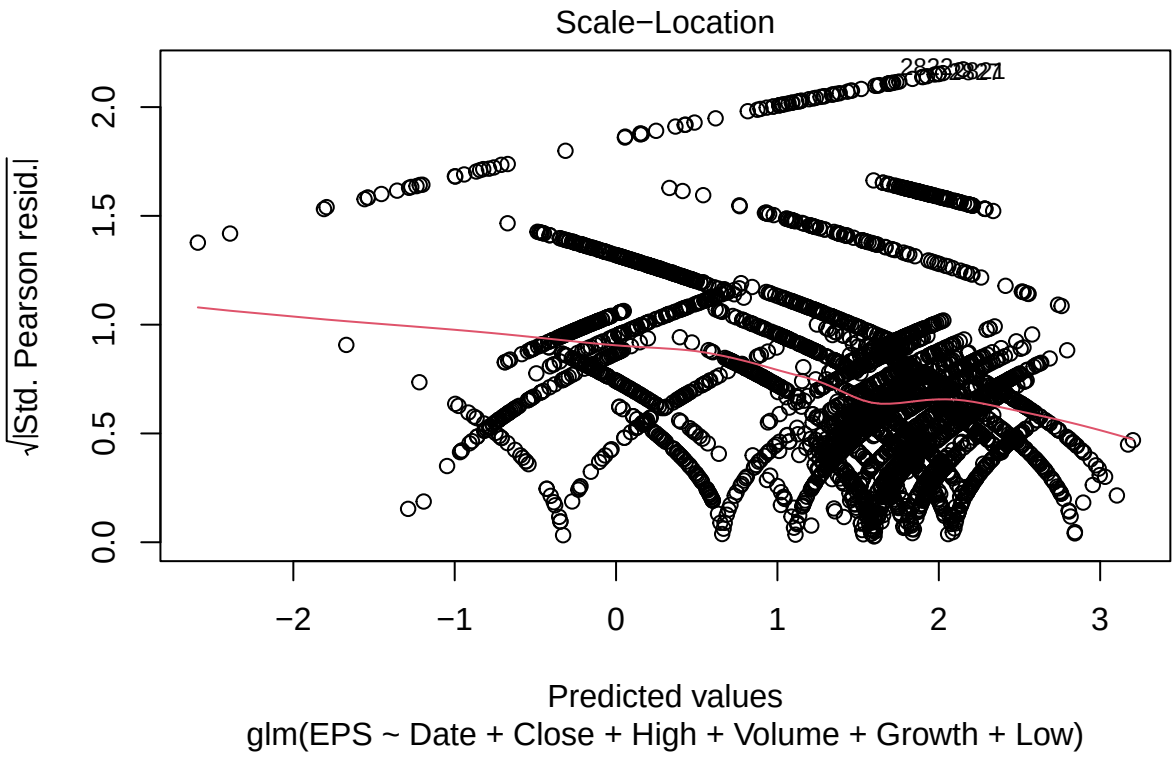
```
mse_glmod_PSX11<- mean((F_df_test_PSX$EPS- predict(glmod_EPS_PSX,
                                                   newdata = F_df_test_PSX,
                                                   type="response"))^2)
mse_glmod_PSX11
```

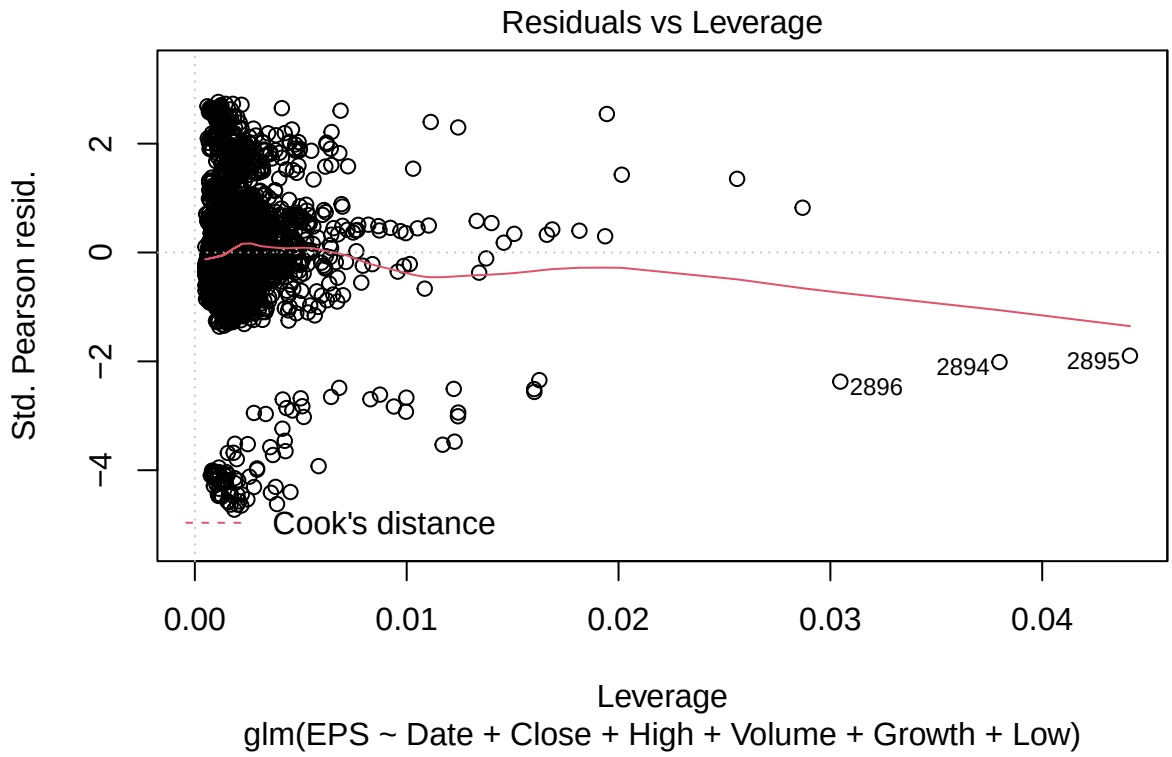
```
## [1] 5.504887
```

```
plot(glmod_EPS_PSX)
```


Residuals vs Fitted
Residuals
6
4
2
0
−2
−4
−6
−8
−2
−1
0
1
2
3
Predicted values
glm(EPS ~ Date + Close + High + Volume + Growth + Low)


Normal Q−Q
Std. Pearson resid.
2
0
−2
−4
−2
0
2
Theoretical Quantiles
glm(EPS ~ Date + Close + High + Volume + Growth + Low)


Scale−Location
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
2.0
−2
−1
0
1
2
3
Predicted values
glm(EPS ~ Date + Close + High + Volume + Growth + Low)



The above value is the Mean Square error for the second model which is 5.504887.

```
glmod_EPS_PSX1 <- glm(EPS~Date+Close+High+Volume,
                  data = F_df_train_PSX, family = 'gaussian' )
summary(glmod_EPS_PSX1)
```

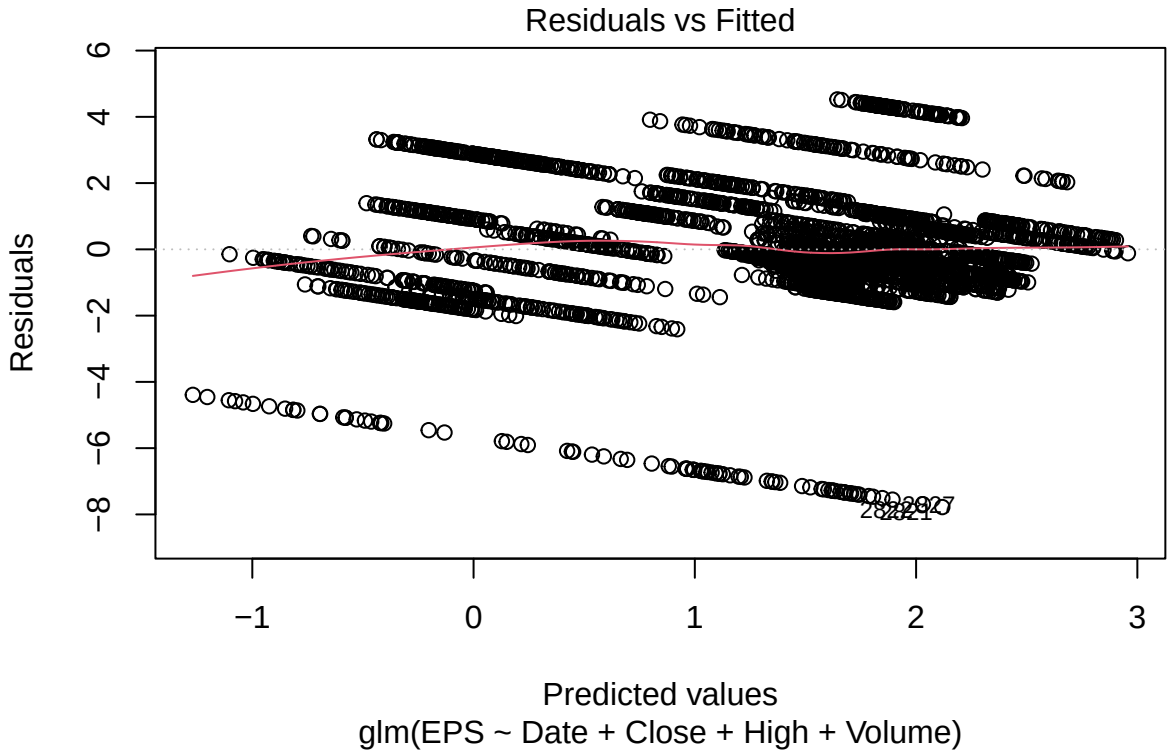
```
##
## Call:
## glm(formula = EPS ~ Date + Close + High + Volume, family = "gaussian",
##     data = F_df_train_PSX)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.7785  -0.7200  -0.1747   0.6395   4.5258
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.105e+01  4.890e-01  22.587  < 2e-16 ***
## Date        -7.463e-04  3.021e-05 -24.703  < 2e-16 ***
## Close        8.806e-02  2.839e-02   3.102  0.00194 **
## High        -4.643e-02  2.829e-02  -1.642  0.10077
## Volume      -5.965e-09  1.856e-08  -0.321  0.74790
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.768562)
```

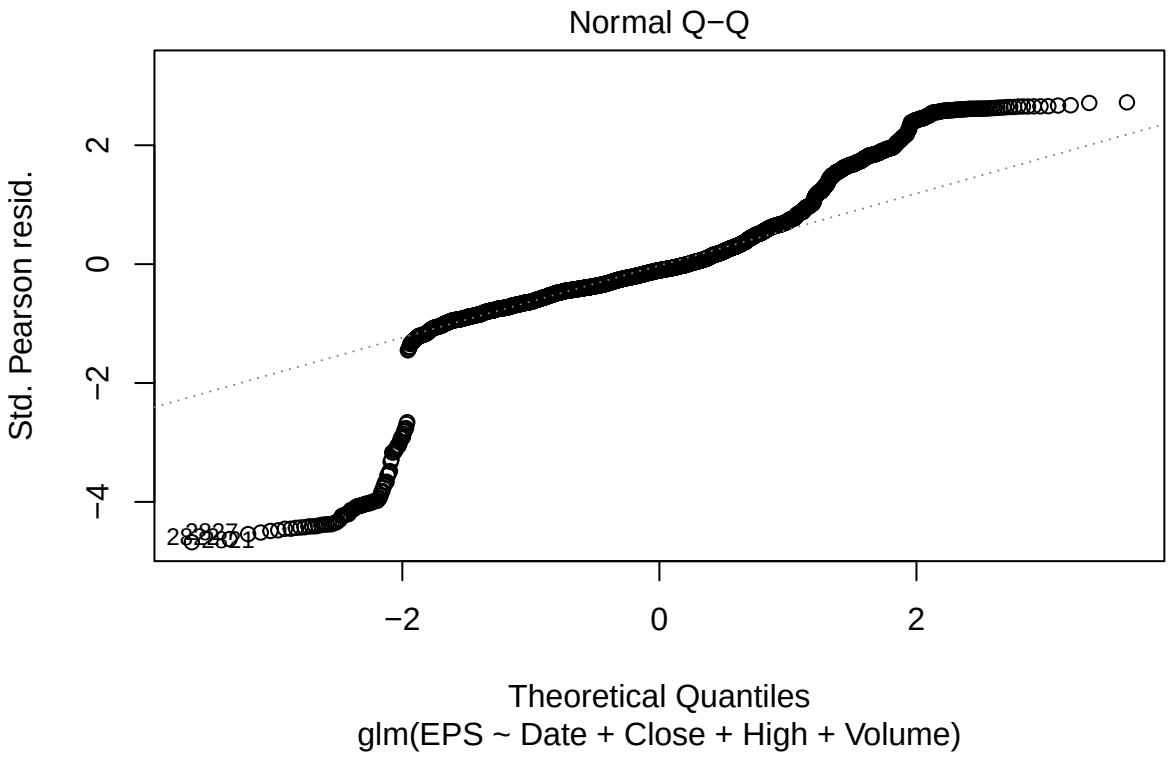
```
##
##     Null deviance: 12482  on 3633  degrees of freedom
## Residual deviance: 10047  on 3629  degrees of freedom
## AIC: 14020
##
## Number of Fisher Scoring iterations: 2
```

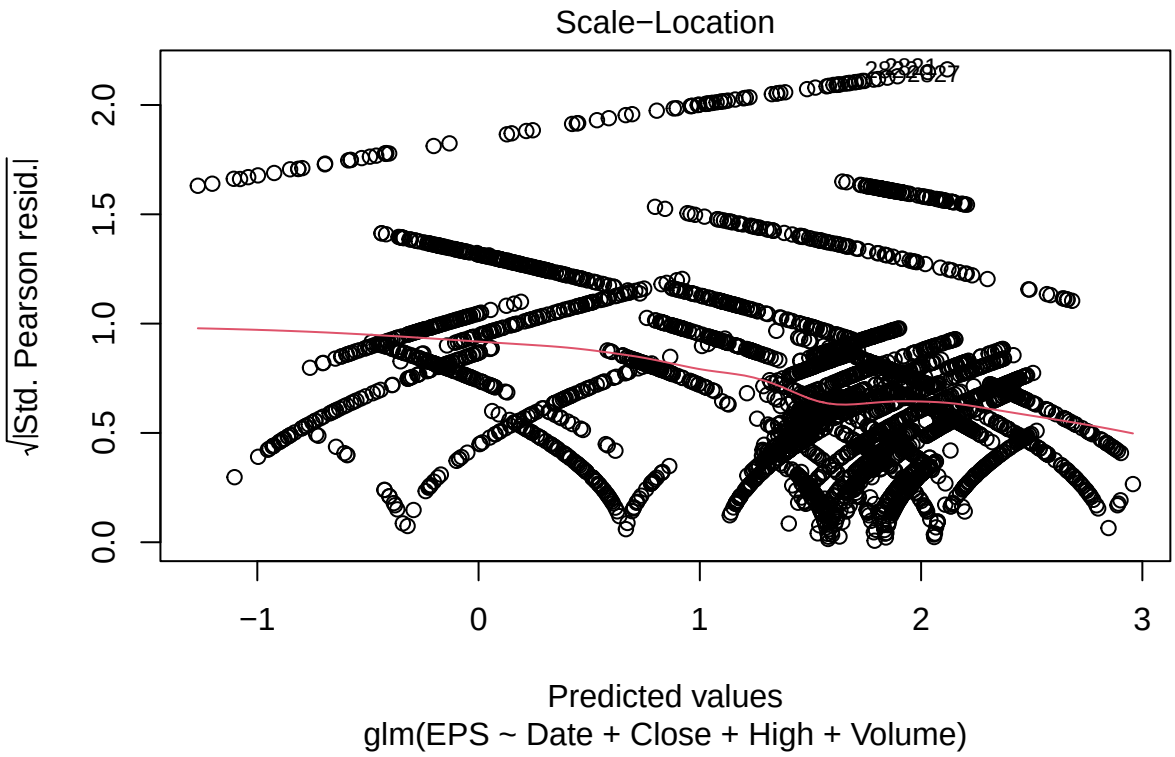
```
mse_glmod_PSX12<- mean((F_df_test_PSX$EPS- predict(glmod_EPS_PSX1,
                                                   newdata = F_df_test_PSX,
                                                   type="response"))^2)
mse_glmod_PSX12
```

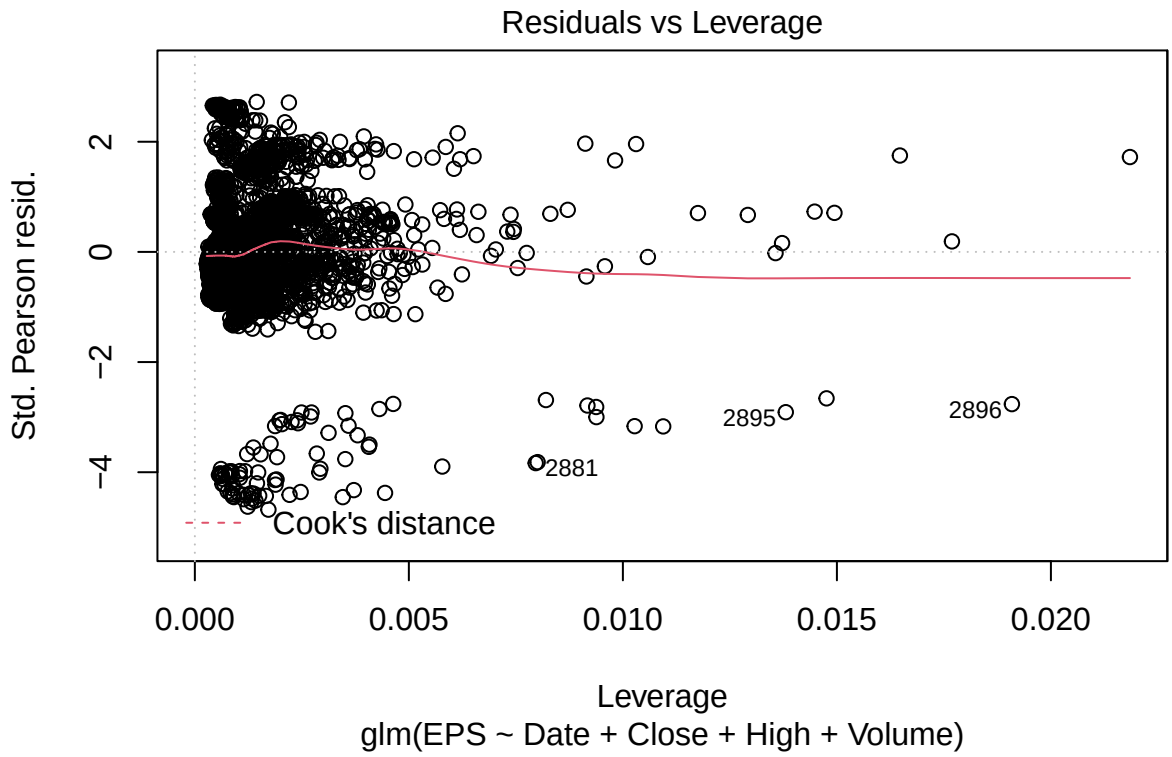
```
## [1] 5.665055
```

```
plot(glmod_EPS_PSX1)
```




Normal Q−Q
Std. Pearson resid.
2
0
−2
−4
−2
0
2
Theoretical Quantiles
glm(EPS ~ Date + Close + High + Volume)


Scale−Location
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
2.0
−1
0
1
2
3
Predicted values
glm(EPS ~ Date + Close + High + Volume)



The above value is the Mean Square error for the second model which is 5.665055.

```
anova(glmod_EPS_PSX,glmod_EPS_PSX1, test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: EPS ~ Date + Close + High + Volume + Growth + Low
## Model 2: EPS ~ Date + Close + High + Volume
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3627     9935.5
## 2      3629    10047.1 -2  -111.64 1.415e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on this anova Chisq test, smaller model is better than the bigger model.

##6.5 A Model With Correlated Random Effects

As EPS changes for quarter, I assumed that is as longitudinal data. Mixed effects model works better for longitudinal dataset.

```
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 4.1.3
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
mixed_model <- glmer(EPS ~ Date +Volume+ Close+(Date | Company_Name), df1_train, family = gaussian(),
          control = lmerControl(optimizer ="Nelder_Mead"))
```

```
## Warning in glmer(EPS ~ Date + Volume + Close + (Date | Company_Name),
## df1_train, : please use glmerControl() instead of lmerControl()
```

```
## Warning in glmer(EPS ~ Date + Volume + Close + (Date | Company_Name),
## df1_train, : calling glmer() with family=gaussian (identity link) as a shortcut
## to lmer() is deprecated; please call lmer() directly
```

```
## Warning: Some predictor variables are on very different scales: consider
## rescaling
```

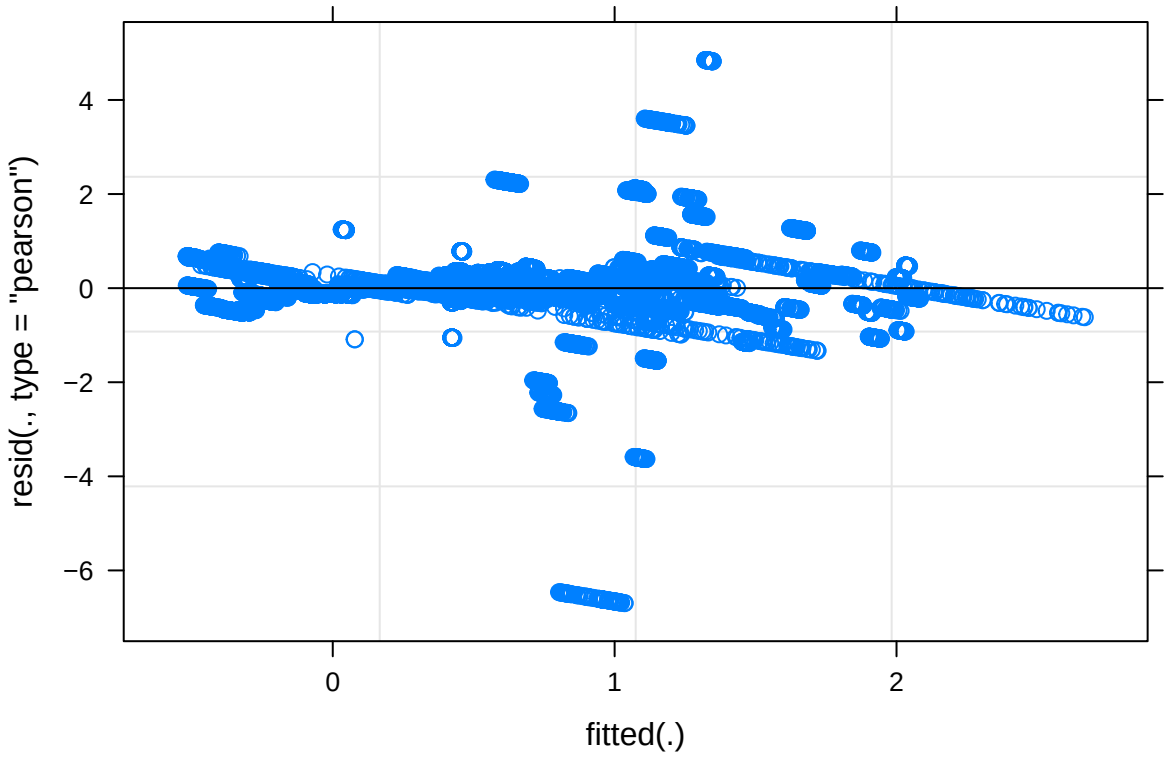
```
## boundary (singular) fit: see help('isSingular')
```

```
summary(mixed_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: EPS ~ Date + Volume + Close + (Date | Company_Name)
##    Data: df1_train
## Control: lmerControl(optimizer = "Nelder_Mead")
##
## REML criterion at convergence: 60889.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.9830 -0.1862 -0.0361  0.2028  6.5049
##
## Random effects:
##  Groups       Name        Variance  Std.Dev.  Corr
##  Company_Name (Intercept) 5.986e+00 2.4465860
##               Date        1.992e-08 0.0001411 -0.98
##  Residual                 5.556e-01 0.7454112
## Number of obs: 26989, groups:  Company_Name, 6
##
## Fixed effects:
##               Estimate Std. Error t value
## (Intercept)  1.375e+00  1.001e+00   1.373
## Date        -5.166e-05  5.778e-05  -0.894
## Volume       1.154e-10  6.547e-11   1.763
## Close        2.790e-03  5.993e-05  46.545
##
## Correlation of Fixed Effects:
##        (Intr) Date   Volume
```

```
## Date   -0.975
## Volume -0.024  0.021
## Close   0.029 -0.034  0.007
## fit warnings:
## Some predictor variables are on very different scales: consider rescaling
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
plot(mixed_model)
```




```
anova(mixed_model)
```

```
## Analysis of Variance Table
##        npar  Sum Sq Mean Sq   F value
## Date      1    0.23    0.23    0.4118
## Volume    1    1.14    1.14    2.0505
## Close     1 1203.77 1203.77 2166.4735
```

```
library(performance)
```

```
## Warning: package 'performance' was built under R version 4.1.3
```

```
##
## Attaching package: 'performance'
```

```
## The following objects are masked from 'package:yardstick':
##
##     mae, rmse
```

```
performance::model_performance(mixed_model)
```

```
## # Indices of model performance
##
## AIC       |      AICc |       BIC | R2 (cond.) | R2 (marg.) |   ICC |  RMSE | Sigma
## ----------------------------------------------------------------------------------
## 60905.359 | 60905.364 | 60970.984 |      0.422 |      0.082 | 0.370 | 0.745 | 0.745
```

```
mse_mixed_model<- mean((df1_test$EPS - predict(mixed_model, newdata = df1_test, type="response"))^2)
mse_mixed_model
```

```
## [1] 0.9124601
```

However, the above model is not very significant as it has more mean squared error.

To my understanding by evaluating all these model I find that GLMs are more effective and interpretable.

#CONCLUSION

In this project I have analized whether or not the stock price has grew in a day and EPS(Earnings per share) which is the earnings per share of the company for every quarter. I have created models for these two targets and identified which model is best. As Phillips66 is an company in energy sector, it is most hiked sector after the conflict and therefore I have considered for modelling.

#REFERENCES

###1. https://www.kaggle.com/code/elenapetrova/time-series-analysis-and-forecasts-with-prophet/notebook

###2. https://arulvelkumar.wordpress.com/2017/07/06/confusionmatrix-function-in-r-the-data-contain-levels-not-found-in-the-data/ ###3. https://plotly.com/r/candlestick-charts/ ###4. https://github.com/congnguyen53/StockMarketAnalysis.R/blob/master/FinalReportRmarkdown.Rmd ###5. https://cran.r-project.org/web/packages/lme4/lme4.pdf ###6. https://www.kaggle.com/code/juejuewang/handle-missing-values-in-time-series-for-beginners/report ###7. http://www.css.cornell.edu/faculty/dgr2/_static/files/R_html/explainRegression.html ###8. https://www.r-bloggers.com/2015/01/using-rvest-to-scrape-an-html-table/