Final Project OPR/STA 9750

KWANG HEUM YEON

1. Project Objective

Predict S&P 500 trends by applying various statistic analysis and a machine learning algorithms – linear regression in supervised learning we learned in the STA9750 class at Baruch College.

2. Motivation

To be successful in stock trading, maintaining a firm mindset and abiding by the initial plan when to realize the gain and loss should be the fundamental attitude. However, because we are human the emotion from time to time makes us vulnerable to huge loss irrevocable. To narrow down these deficiencies and uncertainties of the macro-economy, we would like to predict S&P 500 index representing major stocks listed on the US stock exchange. We believe this project is feasible in that the quant trading becomes popular among investment banks in recent years. Lots of individual traders are also developing their own systems when trading cryptocurrencies, stocks, or NFT. For these reasons, we would like to finally upgrade and optimize the model for analyzing stock portfolios using machine learning algorithms.

3. Data Sources

This data comes from our Github repository. We gathered the real-time macroeconomic indexes by using API-yfinance and then will get subsets. The stock price dataset is based on 'Nasdaq.com' (https://www.nasdaq.com/) and 'NYSE.com' (https://www.nyse.com/). Github repository (https://raw.githubusercontent.com/cpasean/R/main/snp.csv)

4. Data Summary

The dataset includes the S&P500 index with ten different macroeconomic indexes between September 2014 and May 2022. The date column does not include weekends when the market is closed.

- Quantitative variable: CPI(Consumer price index), Jclaim(Jobless claim), Int_rate(10 year bond rate), Stock_price(S&P500 Index), Inf(Inflation rate), Choe(Choe volatility), Gold, Oil(Crude Oil), Dollar(Dollar Index)
- Qualitative variable: date ('mm-dd-yyyy')

Data analysis starts from here.

Set-Up

```
# import required libraries

library(rvest)
```

```
## Warning: package 'rvest' was built under R version 4.0.5
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.0.5
```

```
## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1; sf_use_s2() is TRUE
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## Warning: package 'forcats' was built under R version 4.0.5
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()         masks stats::filter()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
```

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
```

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

```
library(tidycensus)
library(dplyr)
library(ggplot2)
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 4.0.5
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.0.5
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

read data from the repository

```
# import each data from the Github repository
# S&P 500 index
snp <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/snp.csv')
```

```r
# Crude oil
oil <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Oil.csv')
```

```r
# The number of weekly jobless claim
jclaim <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Jclaim.csv')
```

```r
# FED Bond rate - 10yr
int_rate <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Int_rate.csv')
```

```r
# Inflation rate
inf <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Inf.csv')
```

```r
# Gold
gold <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Gold.csv')
```

```r
# Dollar index
dollar <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Dollar.csv')
```

```r
# CPI index
cpi <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Cpi.csv')
```

```r
# Cboe volatility
cboe <- read.csv('https://raw.githubusercontent.com/cpasean/R/main/Cboe.csv')
```

Join datasets

```r
# execute inner join for imported datasets

df <- inner_join(snp, oil, by = "Date")
df <- inner_join(df, jclaim, by = "Date")
df <- inner_join(df, int_rate, by = "Date")
df <- inner_join(df, inf, by = "Date")
df <- inner_join(df, gold, by = "Date")
df <- inner_join(df, dollar, by = "Date")
df <- inner_join(df, cpi, by = "Date")
df <- inner_join(df, cboe, by = "Date")
```

Checklist: Data type, Null, and Five-number summary

```r
# check data type and range
str(df)
```

```
## 'data.frame':    1920 obs. of  10 variables:
##  $ Date    : chr  "2014-09-17" "2014-09-18" "2014-09-19" "2014-09-22" ...
##  $ snp     : num  2002 2011 2010 1994 1983 ...
##  $ Oil     : num  94.4 93.1 92.4 91.5 91.6 ...
##  $ Jclaim  : int  288000 288000 288000 295000 295000 295000 295000 295000 290000 290000 ...
##  $ Int_rate: num  2.6 2.63 2.59 2.57 2.54 ...
##  $ Inf.    : num  2.06 2.02 2.03 1.99 2.02 2.03 2.02 1.97 1.95 1.97 ...
##  $ Gold    : num  1234 1226 1215 1217 1221 ...
##  $ Dollar  : num  84.7 84.3 84.8 84.7 84.7 ...
##  $ Cpi     : num  237 237 237 237 237 ...
##  $ Cboe    : num  12.6 12 12.1 13.7 14.9 ...
```

```r
# check Null
any(is.na(df))
```

```
## [1] FALSE
```

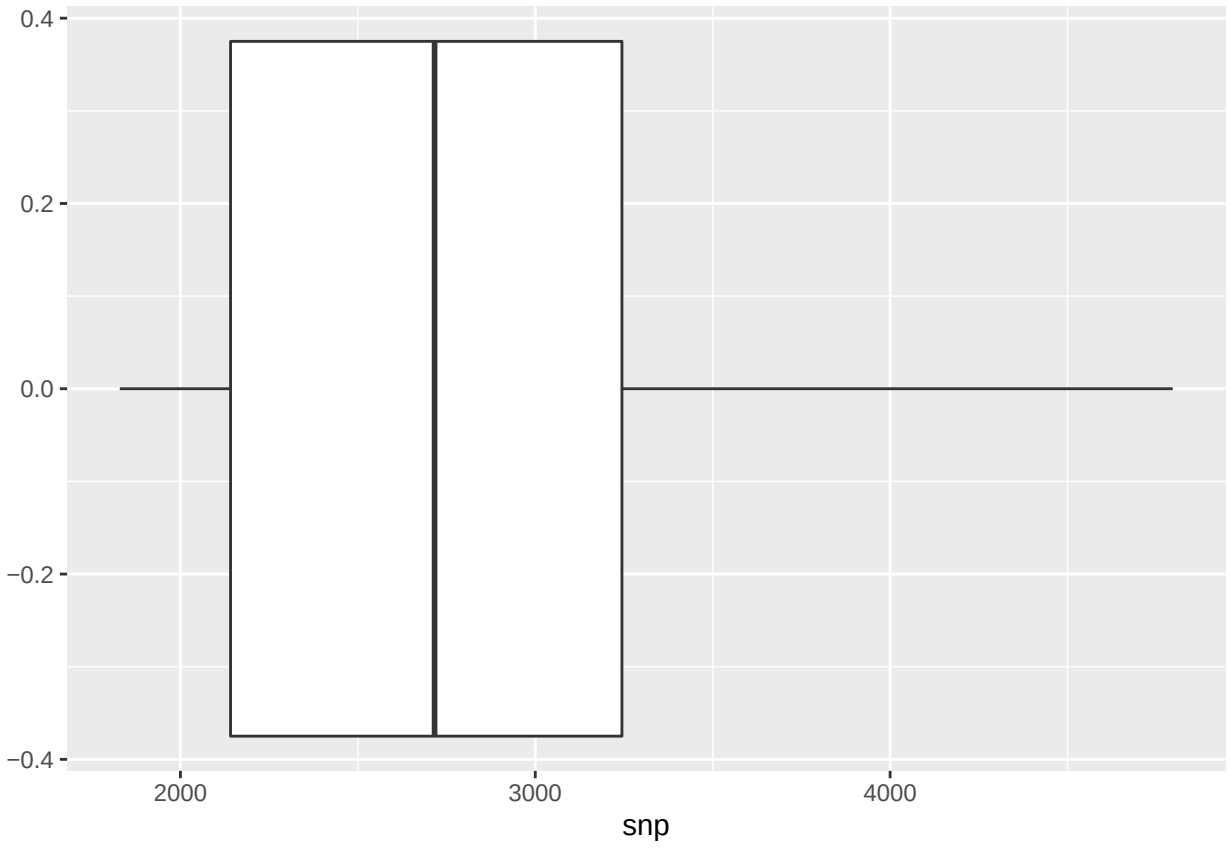
```r
# five-number summary
summary(df)
```

```
##      Date                snp             Oil              Jclaim
##  Length:1920        Min.   :1829   Min.   :-37.63   Min.   : 166000
##  Class :character   1st Qu.:2141   1st Qu.: 46.23   1st Qu.: 226000
##  Mode  :character   Median :2716   Median : 53.55   Median : 256000
##                     Mean   :2864   Mean   : 56.04   Mean   : 419154
##                     3rd Qu.:3244   3rd Qu.: 64.32   3rd Qu.: 291000
##                     Max.   :4797   Max.   :123.70   Max.   :6137000
##     Int_rate          Inf.            Gold          Dollar
##  Min.   :0.499   Min.   :0.500   Min.   :1051   Min.   : 84.32
##  1st Qu.:1.580   1st Qu.:1.630   1st Qu.:1224   1st Qu.: 93.26
##  Median :2.054   Median :1.840   Median :1302   Median : 95.77
##  Mean   :1.975   Mean   :1.874   Mean   :1428   Mean   : 95.37
##  3rd Qu.:2.390   3rd Qu.:2.090   3rd Qu.:1724   3rd Qu.: 97.59
##  Max.   :3.234   Max.   :3.020   Max.   :2052   Max.   :103.87
##       Cpi             Cboe
##  Min.   :234.7   Min.   : 9.14
##  1st Qu.:240.5   1st Qu.:13.02
##  Median :251.3   Median :15.98
##  Mean   :252.3   Mean   :18.07
##  3rd Qu.:258.7   3rd Qu.:21.04
##  Max.   :287.7   Max.   :82.69
```

```r
# table name
names(df)
```

```
##  [1] "Date"     "snp"      "Oil"      "Jclaim"   "Int_rate" "Inf."
##  [7] "Gold"     "Dollar"   "Cpi"      "Cboe"
```

```r
# box plot
ggplot(aes(x = snp), data = df) +
  geom_boxplot()
```

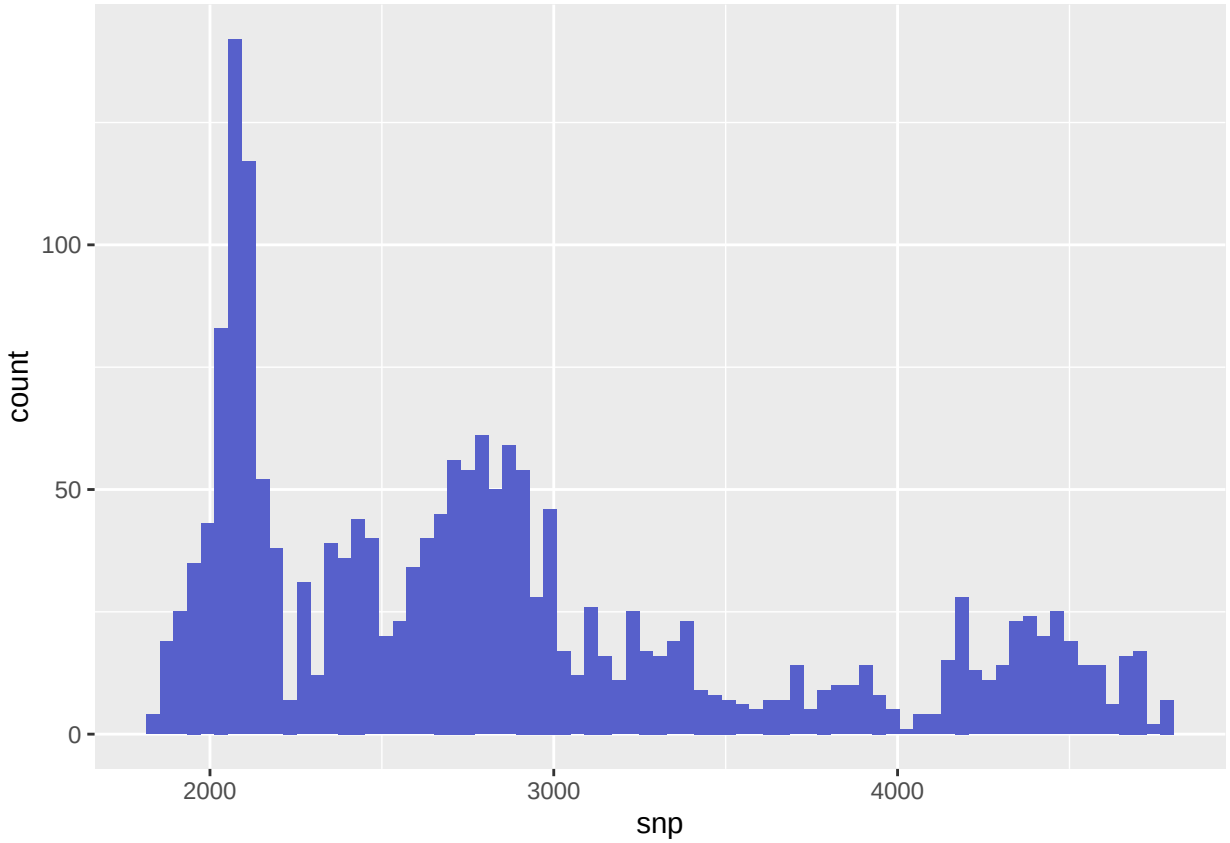


The prediction, S&P 500, forms a right-skewed pattern where the mode is located on the left side.

```
summary(df$snp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1829    2141    2716    2864    3244    4797
```

```
ggplot(aes(x = snp), data = df) +
  geom_histogram(binwidth = sd(df$snp)/20, fill = '#5760CB')
```



Variable name, snp, is the target column so we got a boxplot and histogram and it has a right-skewed pattern having mode and mean at 2716 and 2864 respectively.

Now, we can adjust the bin on the ShinyApp web below

```
library(shiny)
```

```
## Warning: package 'shiny' was built under R version 4.0.5
```

```
##
## Attaching package: 'shiny'
```

```
## The following object is masked from 'package:jsonlite':
##
##     validate
```

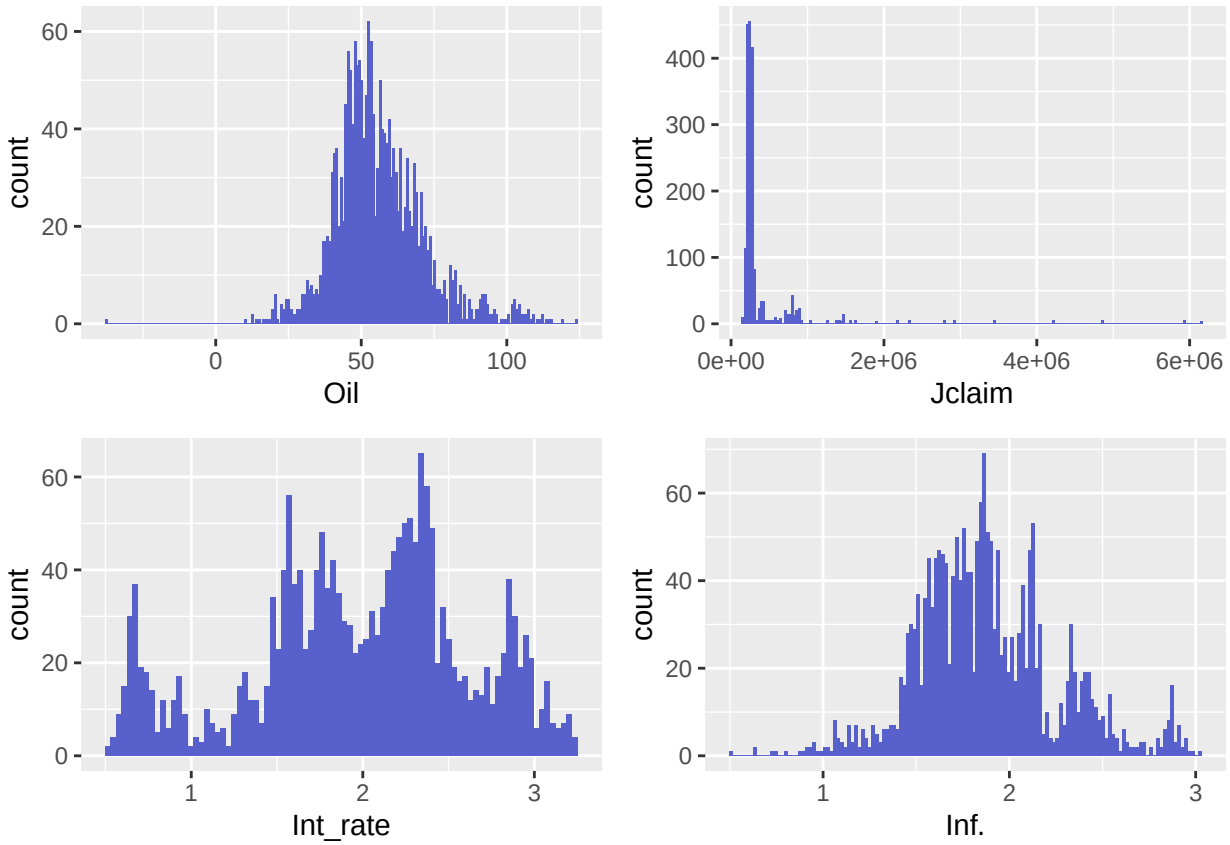
```
# Define UI for app that draws a histogram

ui <- fluidPage(
  titlePanel("Final Project: Predict S&P500 Index"),
  sidebarLayout(
    sidebarPanel(
      # Input: Slider for the number of bins
      sliderInput(inputId = "bins",
```

```
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
      ),
    mainPanel(
      # Output: Histogram
      plotOutput(outputId = "distPlot")
    )
  )
)

# Define server logic required to draw a histogram ----
server <- function(input, output) {

  output$distPlot <- renderPlot({
    x <- df$snp
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    hist(x, breaks = bins, col = "#75AADB", border = "white",
         xlab = "S&P 500 Index",
         main = "Histogram of S&P 500 Index")
    })
}

shinyApp(ui = ui, server = server)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

histogram for each variable

```
# set each bin proportional to the standard deviation
# we adjusted the bin by dividing Std. by 20

p1 <-ggplot(aes(x = Oil), data = df) +
  geom_histogram(binwidth = sd(df$Oil)/20, fill = '#5760CB')

p2 <-ggplot(aes(x = Jclaim), data = df) +
  geom_histogram(binwidth = sd(df$Jclaim)/20, fill = '#5760CB')

p3 <-ggplot(aes(x = Int_rate), data = df) +
  geom_histogram(binwidth = sd(df$Int_rate)/20, fill = '#5760CB')

p4 <- ggplot(aes(x = Inf.), data = df) +
  geom_histogram(binwidth = sd(df$Inf.)/20, fill = '#5760CB')

grid.arrange(p1, p2, p3, p4, ncol = 2)
```

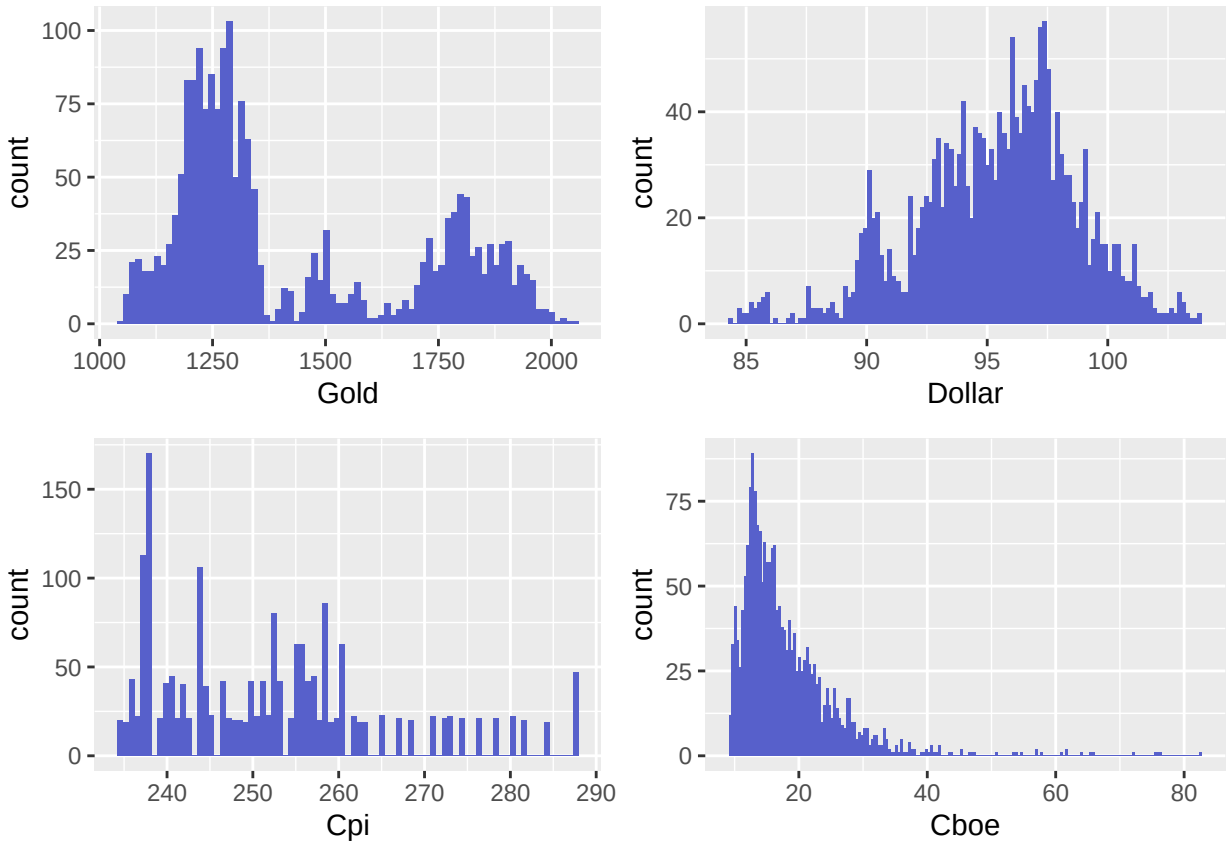


Jobless claim counts, has a right-skewed pattern.

Oil and the Inflation rate has a mode placed approximately at the center.

Intereset rate has two modes on either side from the center and another two small at the edge of the range. The highest mode locates on the left side of the center.

```
p1 <-ggplot(aes(x = Gold), data = df) +
  geom_histogram(binwidth = sd(df$Gold)/20, fill = '#5760CB')

p2 <-ggplot(aes(x = Dollar), data = df) +
  geom_histogram(binwidth = sd(df$Dollar)/20, fill = '#5760CB')

p3 <-ggplot(aes(x = Cpi), data = df) +
  geom_histogram(binwidth = sd(df$Cpi)/20, fill = '#5760CB')

p4 <-ggplot(aes(x = Cboe), data = df) +
  geom_histogram(binwidth = sd(df$Cboe)/20, fill = '#5760CB')

grid.arrange(p1, p2, p3, p4, ncol = 2)
```



CPI and Cboe have a right-skewed pattern.

Dollar index has a slightly left-skewed pattern.

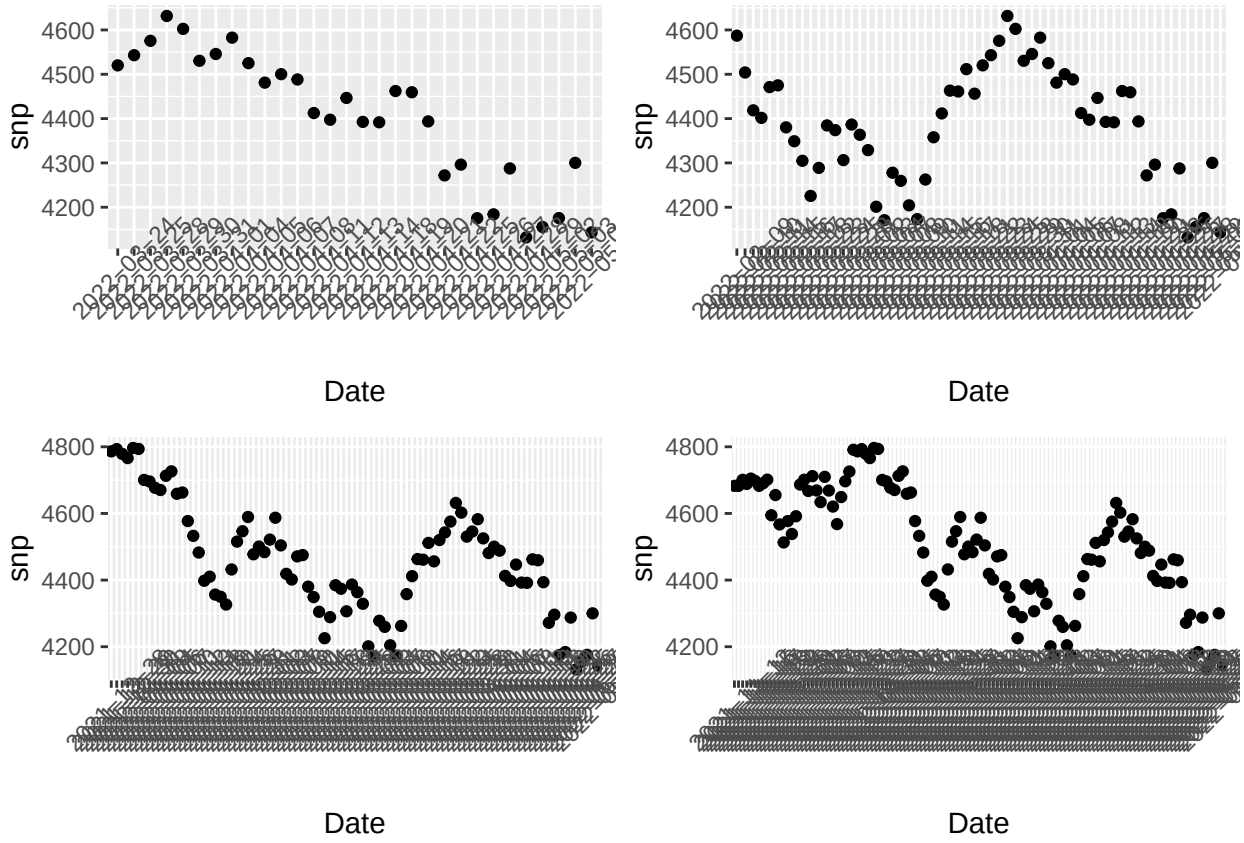
Gold has two modes. The highest mode locates on the left side of the center.

S&P 500 average analysis for 30, 60, 90, and 120 days

```
df_30 = df %>% tail(30)
df_60 = df %>% tail(60)
df_90 = df %>% tail(90)
df_120 = df %>% tail(120)
```

```
p1 <- qplot(x = Date, y = snp, data = df_30) +
  theme(axis.text.x = element_text(angle = 45))

p2 <- qplot(x = Date, y = snp, data = df_60) +
  theme(axis.text.x = element_text(angle = 45))

p3 <- qplot(x = Date, y = snp, data = df_90) +
  theme(axis.text.x = element_text(angle = 45))

p4 <- qplot(x = Date, y = snp, data = df_120) +
```

```
  theme(axis.text.x = element_text(angle = 45))

grid.arrange(p1, p2, p3, p4, nrow = 2, ncol = 2)
```



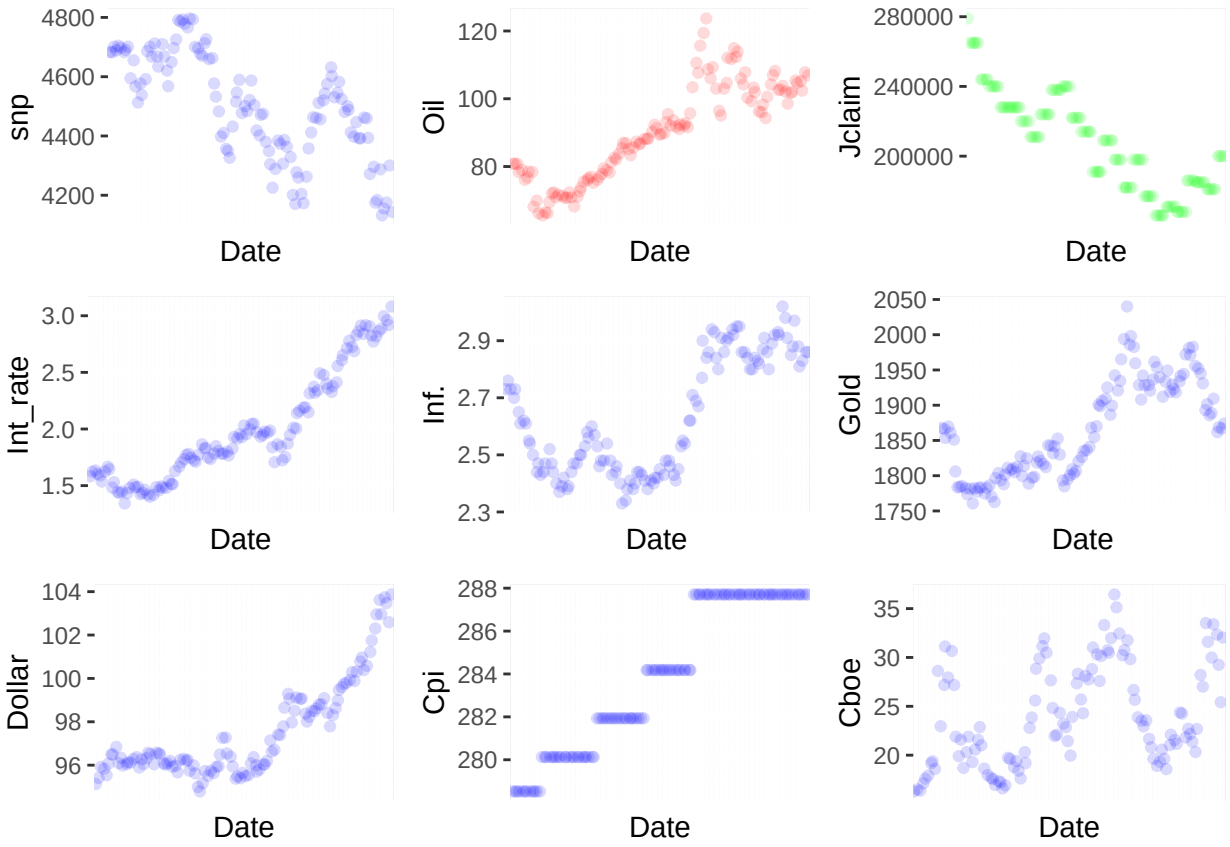
Looking at the 30days graph, the index is going down for the entire range.

There were a huge swing of the index during the past 120 days.

Historical record of all variables for the past 120 days.

```
p1 <- ggplot(aes(Date, snp), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p2 <- ggplot(aes(Date, Oil), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'red') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())
```

```
p3 <- ggplot(aes(Date, Jclaim), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'green') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p4 <- ggplot(aes(Date, Int_rate), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p5 <- ggplot(aes(Date, Inf.), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p6 <- ggplot(aes(Date, Gold), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p7 <- ggplot(aes(Date, Dollar), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p8 <- ggplot(aes(Date, Cpi), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

p9 <- ggplot(aes(Date, Cboe), data = df_120) +
  geom_point(alpha = 0.15,
             position = position_jitter(h = 0),
             color = 'blue') +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, nrow = 3, ncol = 3)
```



We can estimate that S&P 500 index has an inverse relationship with the oil price, interest rate, inflation rate, gold price, the dollar index, and customer price index(CPI)

Before moving on to the machine learning algorithm, we want to understand how closely correlated each variable is. so, we will get the correlation coefficient matrix.

```
names(df)
```

```
##  [1] "Date"     "snp"      "Oil"      "Jclaim"   "Int_rate" "Inf."
##  [7] "Gold"     "Dollar"   "Cpi"      "Cboe"
```

```
# select wanted variable as a subset
df_subset <- df[, c(2:10)]
names(df_subset)
```

```
## [1] "snp"      "Oil"      "Jclaim"   "Int_rate" "Inf."     "Gold"     "Dollar"
## [8] "Cpi"      "Cboe"
```

```
# correlation coefficient matrix #1
cor(df_subset, method = 'pearson')
```

```
##                snp        Oil      Jclaim    Int_rate       Inf.       Gold
## snp      1.00000000  0.5254274  0.09106924 -0.35692175  0.6479808  0.8737461
## Oil      0.52542744  1.0000000 -0.40017375  0.38119380  0.8376569  0.2475428
```

```
## Jclaim    0.09106924 -0.4001738  1.00000000 -0.52718411 -0.3377013  0.3422153
## Int_rate -0.35692175  0.3811938 -0.52718411  1.00000000  0.3136674 -0.6799179
## Inf.      0.64798078  0.8376569 -0.33770134  0.31366736  1.0000000  0.3489303
## Gold      0.87374614  0.2475428  0.34221528 -0.67991794  0.3489303  1.0000000
## Dollar   -0.14688677 -0.2690509  0.08497711  0.04783561 -0.2423993 -0.1643458
## Cpi       0.96800552  0.5425647  0.12262771 -0.30423477  0.6088144  0.8599336
## Cboe      0.24060487 -0.1986392  0.55659283 -0.50206133 -0.2143147  0.4747673
##               Dollar         Cpi        Cboe
## snp      -0.14688677  0.96800552  0.24060487
## Oil      -0.26905095  0.54256469 -0.19863919
## Jclaim    0.08497711  0.12262771  0.55659283
## Int_rate  0.04783561 -0.30423477 -0.50206133
## Inf.     -0.24239929  0.60881437 -0.21431473
## Gold     -0.16434577  0.85993356  0.47476727
## Dollar    1.00000000 -0.01929979  0.06658332
## Cpi      -0.01929979  1.00000000  0.34354764
## Cboe      0.06658332  0.34354764  1.00000000
```

```
# correlation coefficient matrix #2

# facet grid
library(GGally)
```

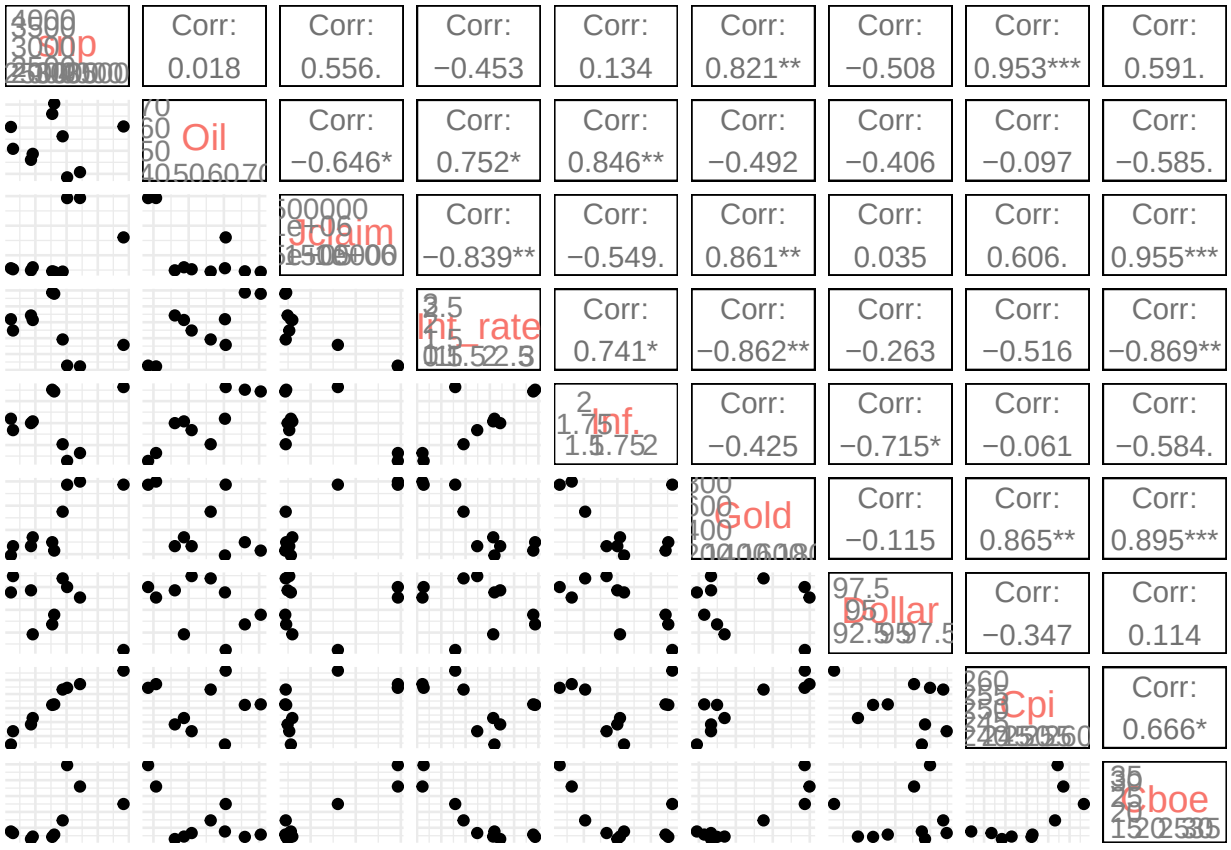
```
## Warning: package 'GGally' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
theme_set(theme_minimal(10))

# set the seed for reproducible results
set.seed(1836)
names(df_subset)
```

```
## [1] "snp"      "Oil"      "Jclaim"   "Int_rate" "Inf."     "Gold"     "Dollar"
## [8] "Cpi"      "Cboe"
```

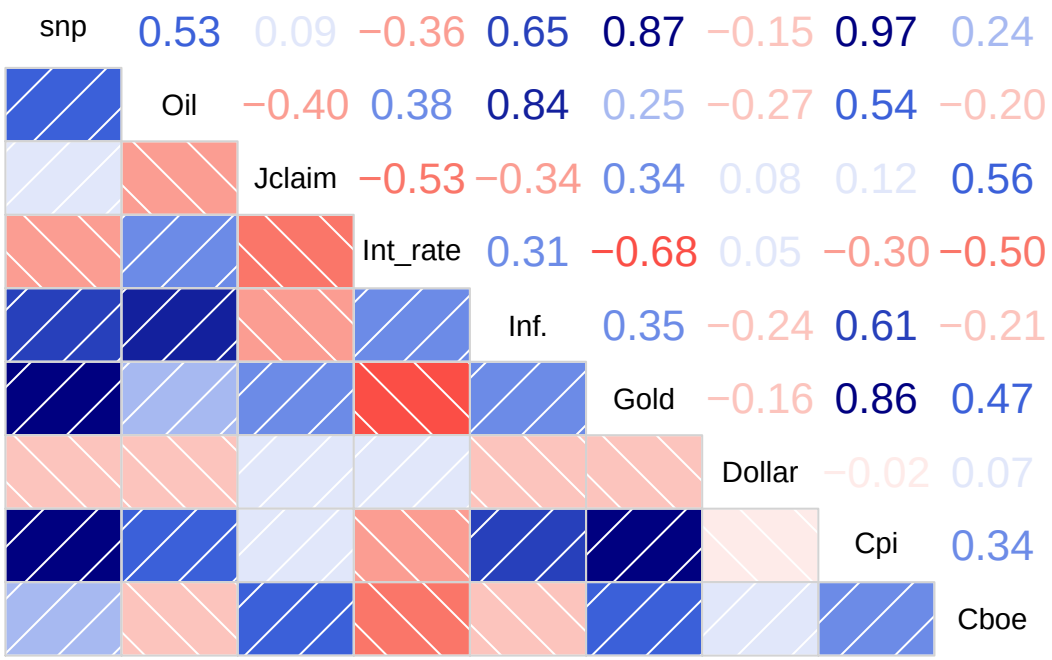
```
ggpairs(df_subset[sample.int(nrow(df_subset), 10), ], axisLabels = 'internal')
```




```
# correlation coefficient matrix #3

library(corrgram)
```

```
## Warning: package 'corrgram' was built under R version 4.0.5
```

```
corrgram(df, lower.panel=panel.shade, upper.panel=panel.cor)
```

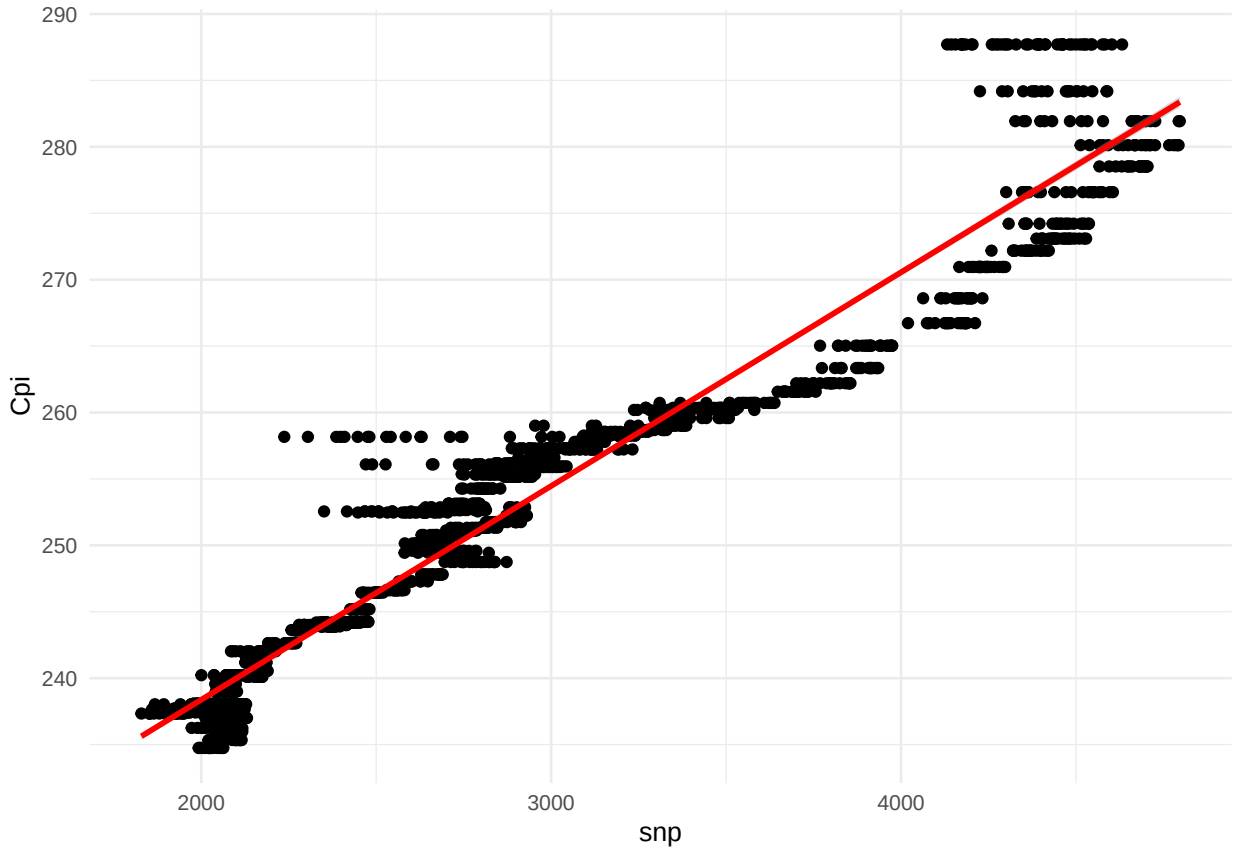


S&P 500 has the highest correlation coefficient with CPI, and the lowest with the jobless claim count. With the interest rate, it's an inverse correlation coefficient.

We will get single linear regression for these three variables.

```
# with CPI
ggplot(aes(x = snp, y = Cpi), data = df_subset) +
  geom_point() +
  geom_smooth(method = 'lm', color = 'red')
```

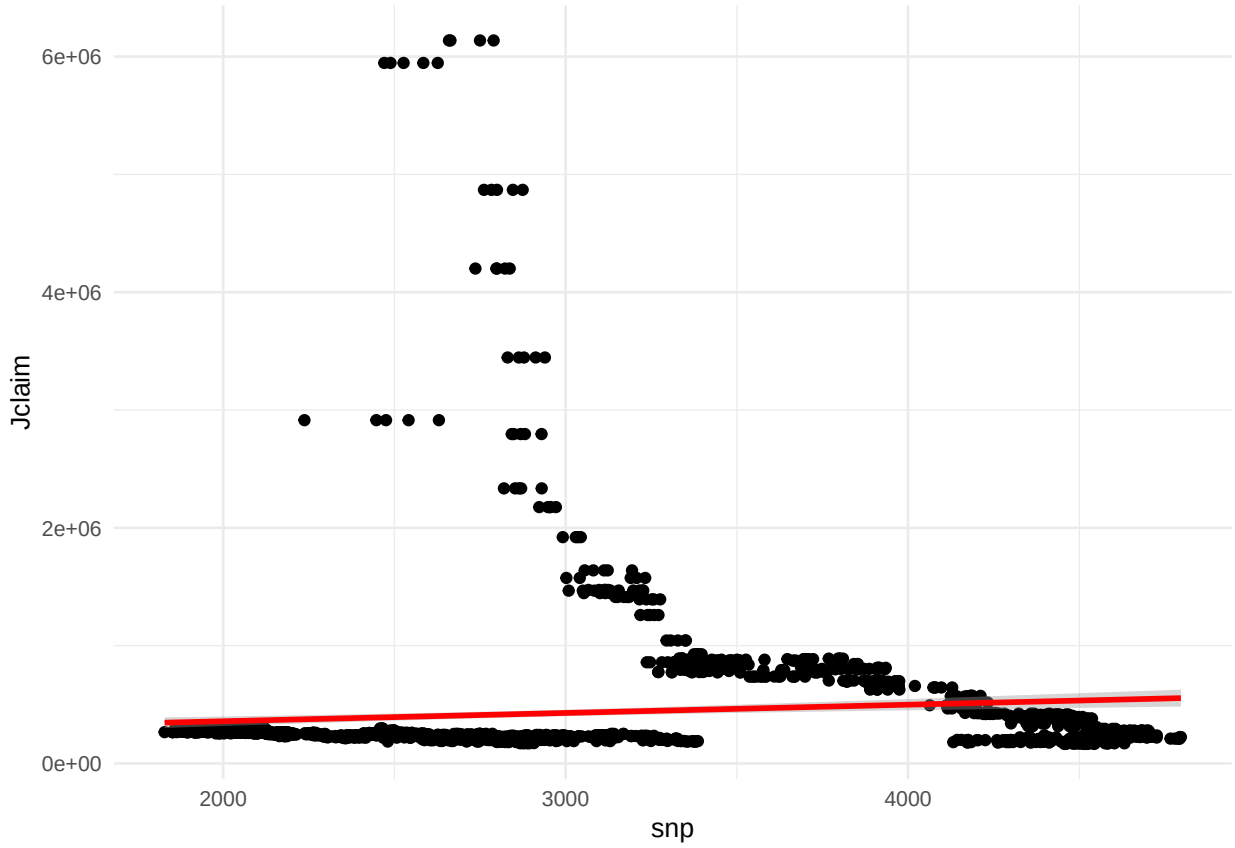
```
## `geom_smooth()` using formula 'y ~ x'
```



S&P 500 index increased as the CPI index increased.

```
# with Jobless claim counts
ggplot(aes(x = snp, y = Jclaim), data = df_subset) +
  geom_point() +
  geom_smooth(method = 'lm', color = 'red')
```

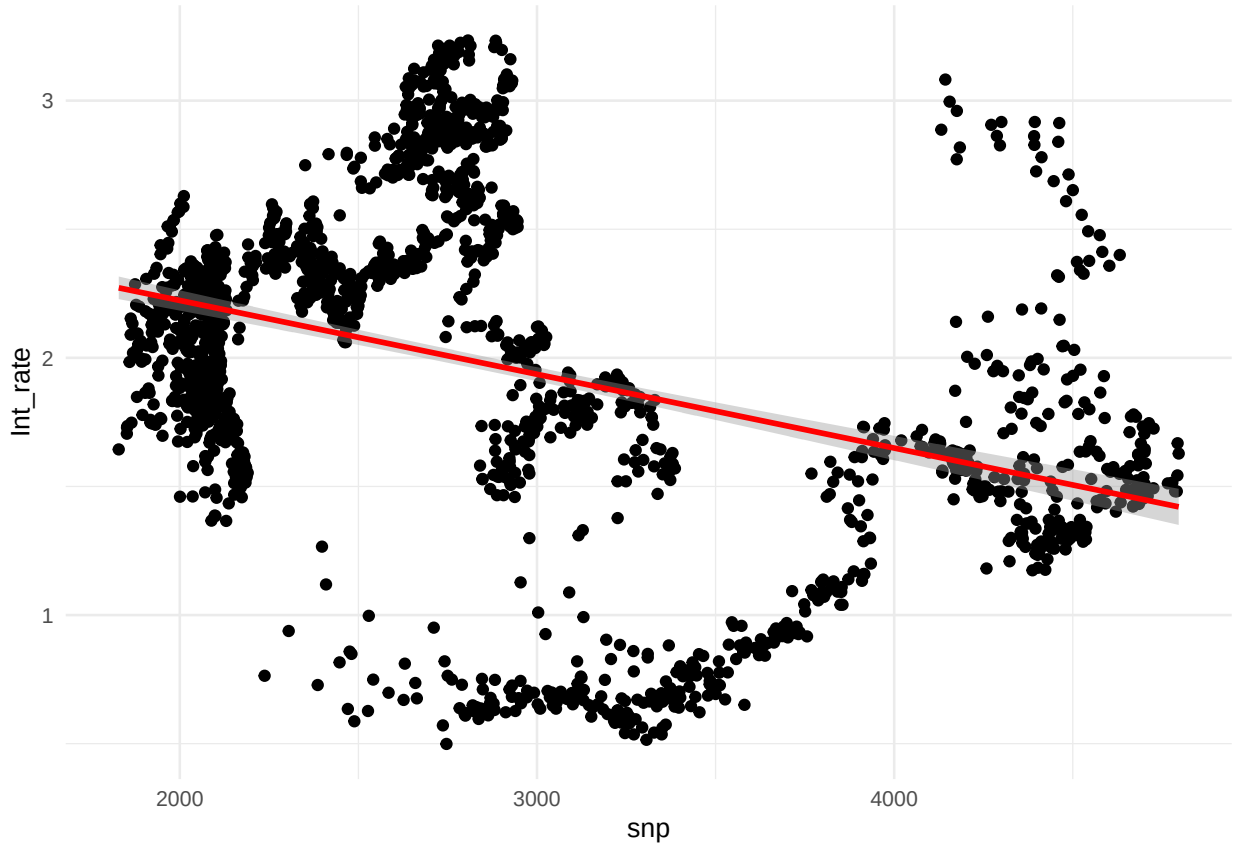
```
## `geom_smooth()` using formula 'y ~ x'
```



S&P 500 index was not significantly affected by the CPI index.

```
# with interest rate
ggplot(aes(x = snp, y = Int_rate), data = df_subset) +
  geom_point() +
  geom_smooth(method = 'lm', color = 'red')
```

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



S&P 500 index decreased as the interest rate increased.

Multi-linear regression

```
summary(lm(df_subset$snp ~
             df$Oil +
             df$Jclaim +
             df$Int_rate +
             df$Inf. +
             df$Gold +
             df$Dollar +
             df$Cpi +
             df$Cboe))
```

```
##
## Call:
## lm(formula = df_subset$snp ~ df$Oil + df$Jclaim + df$Int_rate +
##     df$Inf. + df$Gold + df$Dollar + df$Cpi + df$Cboe)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -672.18  -62.54    5.83   76.07  257.03
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -8.944e+03  9.315e+01 -96.022   <2e-16 ***
## df$Oil      -7.224e+00  3.196e-01 -22.601   <2e-16 ***
## df$Jclaim   -8.955e-06  5.166e-06  -1.733   0.0832 .
## df$Int_rate -2.733e+02  1.003e+01 -27.246   <2e-16 ***
## df$Inf.      4.507e+02  1.442e+01  31.246   <2e-16 ***
## df$Gold     -4.523e-01  4.044e-02 -11.184   <2e-16 ***
## df$Dollar   -2.837e+01  8.912e-01 -31.830   <2e-16 ***
## df$Cpi       6.138e+01  6.576e-01  93.333   <2e-16 ***
## df$Cboe     -1.225e+01  4.243e-01 -28.871   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 103.1 on 1911 degrees of freedom
## Multiple R-squared:  0.9834, Adjusted R-squared:  0.9833
## F-statistic: 1.412e+04 on 8 and 1911 DF,  p-value: < 2.2e-16
```

We conducted a multi-linear regression to analyze the relationship between S&P 500 and the other variables. The R-squared in our model is 0.9834, meaning that 98.34% of the variability in the response - S&P 500 index is explained by the variables such as Oil, Jclaim, Int_rate, Inf., Gold, Dollar, Cpi, and Cboe. through the multi-linear regression model; in other words, the remaining 1.66% of the variability is due to characteristics that are not these variables. Finally, the low p-value explains that the variables, except Jclaim(Jobless claim counts), are statistically significant.

Machine Learning - linear regression

```
# random seed to get the same result
set.seed(42)
```

```
# 70% of the data is used for training, and the remaining 30% is used for testing
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.0.5
```

```
sampleSplit <- sample.split(Y = df$snp, SplitRatio=0.7)
trainSet <- subset(x=df, sampleSplit==TRUE)
testSet <- subset(x=df, sampleSplit==FALSE)
```

```
# model training with linear regression
model <- lm(trainSet$snp ~
              Oil +
              Jclaim +
              Int_rate +
              Inf. +
              Gold +
              Dollar +
              Cpi +
              Cboe, data = trainSet)
```

```r
# summary of our model
summary(model)
```

```
##
## Call:
## lm(formula = trainSet$snp ~ Oil + Jclaim + Int_rate + Inf. +
##     Gold + Dollar + Cpi + Cboe, data = trainSet)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -680.58  -60.08    5.76   75.46  248.03
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.967e+03  1.141e+02 -78.559   <2e-16 ***
## Oil         -7.193e+00  3.955e-01 -18.190   <2e-16 ***
## Jclaim      -1.171e-05  5.619e-06  -2.083   0.0374 *
## Int_rate    -2.698e+02  1.223e+01 -22.058   <2e-16 ***
## Inf.         4.590e+02  1.766e+01  25.988   <2e-16 ***
## Gold        -4.449e-01  4.928e-02  -9.027   <2e-16 ***
## Dollar      -2.787e+01  1.089e+00 -25.584   <2e-16 ***
## Cpi          6.113e+01  7.962e-01  76.770   <2e-16 ***
## Cboe        -1.189e+01  5.064e-01 -23.484   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102.7 on 1335 degrees of freedom
## Multiple R-squared:  0.9831, Adjusted R-squared:  0.983
## F-statistic:  9685 on 8 and 1335 DF,  p-value: < 2.2e-16
```

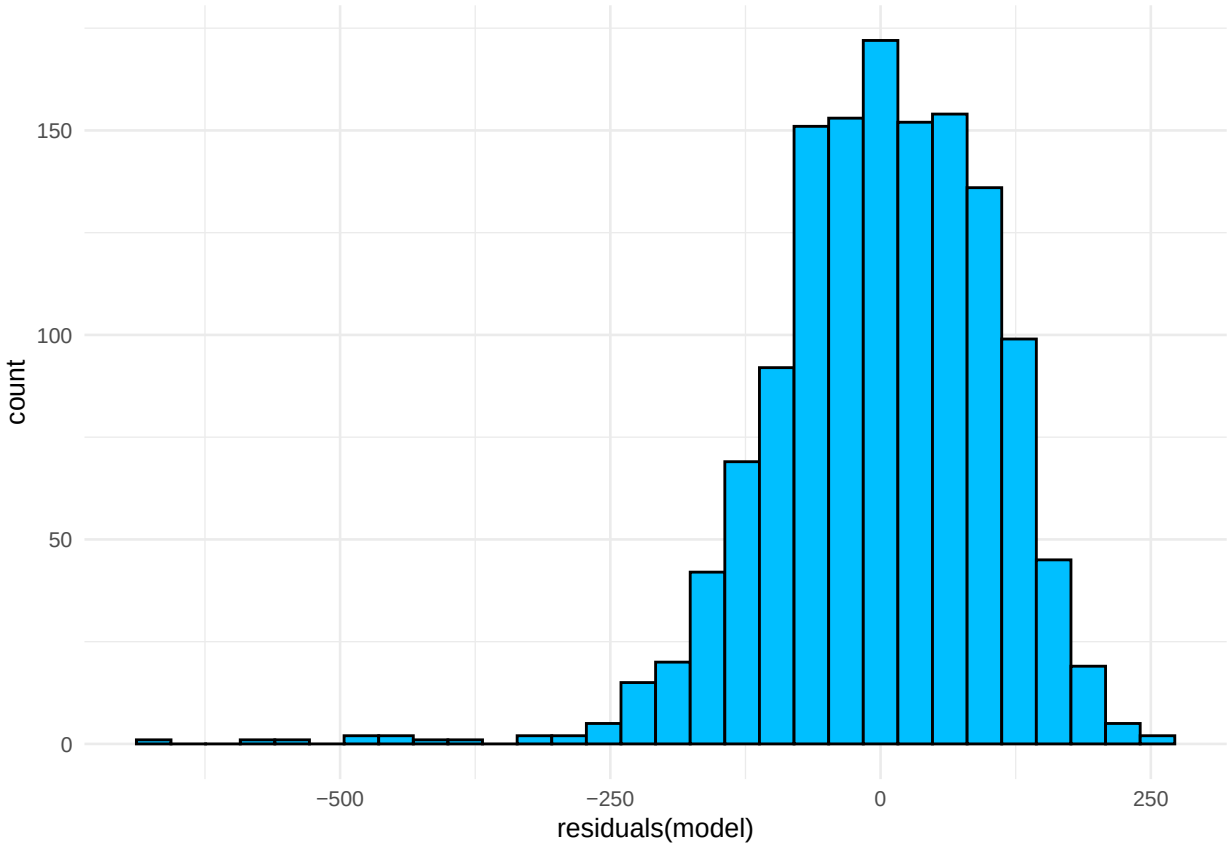
R-squared and p-values of the fitted model are very slightly different from the previous result, which is a meaningful outcome.

```r
modelResiduals <- as.data.frame(residuals(model))
```

Make a residuals histogram to be more precise

```r
ggplot(modelResiduals, aes(residuals(model))) +
  geom_histogram(fill='deepskyblue', color='black')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The histogram is a bit of skew due to the value on the far left, but we can conclude that the residuals are approximately normally distributed.

Make prediction

```
preds <- predict(model, testSet)
```

```
# dataframe for actual and predicted values
modelEval <- cbind(testSet$snp, preds)
colnames(modelEval) <- c('Actual', 'Predicted')
modelEval <- as.data.frame(modelEval)
modelEval %>% head(3)
```

```
##    Actual Predicted
## 1 2001.57  2050.539
## 2 2011.36  2055.901
## 4 1994.29  2044.661
```

Some of rows have a remarkable gap between the actual and predicted value, so we will check the accuracy.

Test the accuracy

```
# get MSE(Mean Square Error)
mse <- mean((modelEval$Actual - modelEval$Predicted)*(modelEval$Actual - modelEval$Predicted))

# get RMSE(Root Mean Square Error)
rmse <- sqrt(mse)
rmse
```

```
## [1] 104.227
```

We got the RMSE value of 105.74 meaning that the average predictions are incorrect by 105.74 units of features.

5. Conclusion

We wanted to predict the S&P500 index based on the given features - Crude oil, the number of weekly jobless claims, 10-year bond rate, inflation rate, inflation rate, gold, the dollar index, CPI index, and Cboe volatility, from the dataset containing the historical record from September 17, 2014. From analysis for the past 120 days, we found that S&P 500 index has an inverse relationship with the oil price, interest rate, inflation rate, gold price, the dollar index, and customer price index(CPI). In addition, S&P 500 has the highest correlation coefficient with CPI and the lowest with the Jobless claim count. With the interest rate, it's an inverse correlation coefficient. S&P 500 index remarkably increased as the CPI index increased when we set the period as the entire range. This is the opposite of the previous 120-days analysis.

We also conducted a multi-linear regression to analyze the relationship between S&P 500 and features. The R-squared in our model is 0.9834, meaning that 98.34% of the variability in the response - S&P 500 index is explained by the features except for Jclaim(Jobless claim counts). Finally, to predict the S&P 500 index, we fitted the training dataset into a machine learning algorithm – linear regression and get the RMSE value of 105.75 meaning that the average predictions are incorrect by 105.74 units of features.

Since the pandemic began, we are suffering from an economic downturn. The US government tried to subsidize people who lost their jobs so the economic stimulus plan has decreased the currency value. The stock market has the highest volatility for the first time in the past decade, the interest rate is increasing and the inflation rate is skyrocketing. The motivation of this project was to understand the economic trend with the analysis and finally cope with the uncertainty by minimizing the potential loss in our investment or financial planning. We learned which macro indexes would positively or negatively affect the S&P 500 index, and how much. With the unexpectedly high r-squared from our analysis, we believe we can keep working on the consecutive research by applying different machine algorithms or additional variables worth putting in.

6. Reference

- Dario Radečić, (2022, December 24), Machine Learning with R: A Complete Guide to Linear Regression, https://www.r-bloggers.com/2020/12/machine-learning-with-r-a-complete-guide-to-linear-regression/
- J.B. MAVERICK, Most important moving average, Investopedia, "Why the 50-Day Simple Moving Average Is Popular Among Traders". https://www.investopedia.com/ask/answers/012815/why-50-simple-moving-average-sma-so-common-traders-and-analysts.asp
- Cory Mitchell (2021, April 28), Useful moving average, Investopedia, "How to Use a Moving Average to Buy Stocks". from https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp

Appendix – Shiny APP

- Variable name: S&P500 - Prediction