Forecasting Gold Prices with Common Indicators

Amin Fesharaki, Ryan Dunn, and Kyle Esteban Dalope

University of San Diego

Master of Science, Applied Data Science

ADS-506 Applied Time Series Analysis

Section 01

December 3, 2022

Introduction and Problem Statement

In this study, the topic that will be investigated and forecasted will be gold prices in the financial market(s). For this time series forecast, the gold prices will be retrieved as daily time series data from November 12, 2012 to November 12, 2022. The prediction of the gold prices forecasted will be evaluated with common external indicators and/or factors that will allow the determination of a relationship that may or may not exist with the price of gold. Such indicators include Money supply, US dollar index, Unemployment rate, Internet web hits of gold, Dow Jones Industrial Average, Silver prices, and Federal funds rate. While gold has maintained its value throughout history, the interest for this project is the intention to research and identify key relationships of gold pricing with the selected external information which will lead indications on the best times to invest in gold for future investors.

As a precious metal, gold has for centuries been a representation of wealth and regarded as a safe haven when economies grow or recess over time; was used as a medium of exchange and became so popular there was an era known as the gold standard of which ended in the 1930s (Chirwa and Odhiambo, 2020). Maintaining this constant trait of being a safe haven, gold has always been a strategic investment asset to be traded in the stock market. Thus, through this investigation into the price dynamics of gold is and will be of great importance to both producers and investors (Chirwa and Odihiambo, 2020). Current trades and indicators for the determinants of gold price movements have been selected, in an attempt to identify what may hinder or drive the price movements in the stock market and economic developments at the daily frequency of the time series.

The aim of this project was to investigate and identify probable time series models with selected external factors to accurately forecast the prices of gold (in USD). As predicting any

data that involves stock market information, generating and developing models that have the ability to have been found to be of chance; which leads to the difficult task of how to handle volatile data such as gold prices with external predictors. With the selected models of Moving average, ARIMA, logistic regression, and Prophet the team agreed upon success criteria of accuracy, RMSE, trend to gold prices, and the calculated profits. This was determined to combat as no two models would be identical when predicting stock prices and allowed each model's tuning parameter(s) utilized appropriately.

Literature Review

Macroeconomic indicators can be an integral part of developing robust forecasting models, as many of these high level indicators can have a substantial impact on the underlying asset price. Inflation, the overall price level in an economy, is a key macroeconomic indicator to assess relative to asset prices, as high inflation can distort incentives for investments (Jones, 2021). Inflation is directly associated with a country's central bank printing money, which can be evaluated directly by the inflation rate, or by other means such as the M2 level (aggregated demand deposits, savings accounts and money market account balances). The M2 level is an important component to understanding inflation because people typically want to hold less money when inflation is high, meaning they go to the bank more often (Jones 2021). Furthermore, in the short-run, society faces a tradeoff between unemployment and inflation. To lower unemployment, policy makers can increase aggregate demand, at the cost of higher inflation. Conversely, if they decrease aggregate demand, inflation may lower, but at the cost of higher unemployment. Professional economists use these variables as leading economic indicators in forecasting models (Jones, 2021).

As the goal for this study was to forecast gold prices with time series analysis methods, the team conducted research to identify the common methods for trading algorithms and indicators that are often used in analysis and evaluation currently. Although forecasting stock price movements, such as gold prices, are predictions that consist of a game of chance due to asset prices being random walks; including external factors and indicators will allow traders better understand the drivers of the prices to support better risk management (Shmueli and Lichtendahl, 2018). The topics of review will include trading algorithms, forecasting gold prices, current methodologies of stock prices, and the common factors that drive or hinder gold prices in the financial markets.

"Algorithmic trading has evolved exponentially in recent years and, along with it, interest in high-frequency trading has grown remarkably; leading to more studies of computational trading algorithms that are useful for investment timing" (Chan and Zainudin, 2016). In the article, *Assessing the Efficacy of Adjustable Moving Averages Using ASEAN-5 Currencies*, the topic of using the time-varying volatility technical analysis indicators Adjustable Moving Average (AMA) will allow the understanding of using AMA to forecast gold prices in this study and the trends that may exist. In this paper in particular, the researchers provided evidence of how AMA can decipher such trends in the exchange rate markets and how moving averages and AMA's are superior in comparison to the passive buy-and-hold strategy. An important factor to consider with financial market trading the authors discussed was the emphasis of correctly identifying the market condition and trading strategy. Chan and Zainudin states "a ranging market requires technical analysis tools (such as leading momentum rate of change) that differ from those employed in a trending market (like lagging moving average)" (2016). The Adjustable Moving Average intends to avoid some of the whipsaws (losses due to wrong trading

signals) and to allow early entry into new trends by employing a time-varying trend deciphering ratio called the Efficacy Ratio. This analysis with currency markets allowed the utilization of moving averages (i.e. AMA) to effectively filter out the noise from random walks such as gold prices as a method to attempt in this study.

While researching the topic of forecasting gold prices, the team initially researched its relationship to forecasting stock market prices to understand the scope and computationally constraints that may be experienced throughout the study. As Omar et al. has stated "Stock market forecasting is considered the most challenging problem to solve for analysts. In the past 2 years, Covid-19 has severely affected stock markets globally, which, in turn, created a great problem for investors" (2022). Thus, considering the time frames of interest for the training and validation periods was discussed as there are many variables and factors that may have affected the prices of gold and if to opt out or rather ignore the prices during the pandemic. In the article, *Stock Market Forecasting Using the Random Forest and Deep Neural Network Models Before and During the COVID-19 Period*, it was able to shed some light on another machine learning model to use to forecast the gold prices; autoregressive deep neural network. Within this article the authors showed a practical implication as they can be used by investors and policy makers in their investment decisions and in formulating financial decisions and policies, respectively (2022).

Methodology (Explanation of Steps)

Data Preprocessing

For this study, all work and processing was conducted through the programming language and environment of R. The selected dataset for the gold prices, *Gold.csv*, has been sourced from Nasdaq which contained 2549 features with six variables which include Date,

Close/Last, Volume, Open, High, and Low; however, only Data and Close/Last were used in this study. As each of the .csv files were imported, the initial steps were to identify any key identifiers, trends, seasonality, and distributions that may have existed through the process of exploratory data analysis. All datasets were downloaded at the 5-year level, covering the periods of November 2017 to November 2022. Table A1 overviews all of the .csv files that contained all of the datasets of interest with a brief description for each along with their sources.

All predictor datasets were left joined onto the Gold.csv file by date; however, not all data was available at the daily level. For some predictor variables, such as the unemployment rate, data is only updated monthly. To address this misalignment in data granularity, the MIN and MAX date of each dataset was captured, and a date object was created which contained all days in between. By using the mutate function and na_locf functions in base R, the most recent value for a given predictor variable was used to fill in all daily date ranges until a new value was found. This enabled the data to be downloaded at various levels and mutated into a daily level that was able to be joined onto the daily gold data. Each data attribute was then normalized for the purposes of comparing trends on a single axis and for potential use in predictive models that require the data be on the same scale.

Exploratory Data Analysis

To assess which variables may have a measurable impact on the closing price of gold, a correlation matrix was developed for all possible predictors. Figure B1, a correlation heatmap displays the moderate and highly correlated predictors to the closing price of gold included Silver_Close at 0.86,DFF (Federal Funds Rate) at -0.72, WM2NS (M2) at 0.88, hits at 0.73, and DJIA (Dow Jones Industrial Average) at 0.70.

To assess similarity or differences in trends, Figure B2 displays the variables that were normalized and plotted on the same y-axis relative to time. The resulting time series trends allowed for a better observable trend for many of the predictor variables, and support the assumption that some economic predictors are positively correlated to the closing price of gold during periods of strong economic growth yet are inversely related during periods of economic slowing. This is best observed by viewing both the normalized close price of gold and the normalized federal funds rate.

Figure B3, scatterplots of each predictor variable and the Gold_Close variable were developed with the inclusion of adding the Date as an overlay. This allowed for not only evaluation of any possible relationships between the outcome variables, but also how these may differ by different time periods. Finally, boxplots were developed for each predictor variable to assess possible outliers. Only unemployment rate, gold volume, and hits predictors had outliers. No outliers were removed from the dataset, however, as these outlier events could have a significant effect on the close price of gold which would need to be captured and modeled.

Discussion and Results

Autoregressive Integrated Moving Average Model (ARIMA)

Parameters for an Autoregressive Integrated Moving Average Model (ARIMA) model were explored, as the ARIMA model can develop a more flexible forecasting method. However, an important first step in assessing if the dataset is compatible with an ARIMA model is by evaluating the ACF plot. If an ACF plot indicates autocorrelation lags at 1, 2, and 3 then the series can be assumed to be a random walk. As this was the case with the gold prediction data, the ARIMA is suggested to not be used.

Logistic Regression Model

A logistic regression model was fitted to the data to obtain the predicted probability for each time period of interest. For this model, a binary attribute was created by assessing whether the daily value increased or decreased relative to the day prior. For days with an increase, a 1 was assigned, and for days with a decrease, a 0 was assigned. The training data was then used to fit a logistic regression model with all predictor variables and tested against the date range of November 1, 2022 to November 17, 2022. Table A2, shows how the performance was assessed for the logistic regression model with a confusion matrix.

The logistic regression model with all predictors correctly predicted 69% of the increases/decreases correctly, with a sensitivity score of 69% and Specificity score of 67%. However, a second logistic regression model was developed which included only the moderately or strongly correlated predictors. This model predicted all 17 days as a 1 (increase). It was expected that the highly correlated predictor model would outperform the full predictor model. As such, despite the logistic regression model with all predictors appearing to perform well, the logistic regression method appears to be too volatile to return consistent predictive results and is suggested to not be used.

Prophet

Facebook's Prophet was used in forecasting time series data where non-linear trends fit yearly, weekly, and daily seasonality. Prophet forecasting works well with time series data that have strong seasonal effects with many seasons of historical data. Prices of gold follow a seasonal cycle where it tends to rise in the first quarter and last months of the year. In addition, the external regressors implemented in the model also follow a seasonal pattern: US Money Supply, DOW Jones Industrial Average, Google Trend Hits, Federal Funds Data, and Silver.

With years of historical data for our target variable and predictors, Prophet forecasting was an ideal candidate for gold price prediction.

Four models were created using Prophet forecasting. The first model served as a baseline model by only using the price of gold, with no external regressors, to predict the price of gold one week in advance. For the rest of the models, a rollover partitioning was used to predict the price of gold one period of forecasting at a time for several weeks with the addition of the external regressors. Rollover partitioning was done to forecast a short period of time using the most recent data. Forecasting a random walk time series months or years in advance can lead to significantly inaccurate results. Moreover, the last model used rollover partitioning as well, but instead using actual historical external regressor data to demonstrate that using accurate external data leads to an accurate forecast of gold prices. Models forecasting in one week in the future will use 5 day lag (5 trading days in a week) for the external regressor data and a 1 day lag for models that forecast a day in advance. For example, last week's external data will be used to predict next week's data. Lagged external regressor data was used in conjunction with rollover partitioning to be able to forecast a week at a time with data that would be available at the time of prediction. This method was done to avoid having to forecast all external data separately to be used in the model.

A forecast plot displaying the model performance on training data, time series decomposition of the model, and a plot of predicted and actual values to better compare the two that were created for each of the models. Additionally, the RMSE was also calculated to compare the performance of each model; however, stock prices can be volatile which can affect the RMSE value. For example, the model can forecast well for one week and terribly the next if the price spikes up or down. Therefore, two other success criterias were implemented for gold price

forecasting due to its random walk nature. The first criteria was determining whether the forecast follows the same trend as the actual gold price. This can be beneficial to investors to use the model to forecast whether gold will increase or decrease in the coming weeks regardless of how accurate the prediction is price wise. More importantly, the second success criteria will be whether or not an investor will make a profit following the model. To measure this, a function was created to determine the local minimum and maximums of the predicted price points and assign a buy or sell signal on the date (i.e. minimum is a buy signal, maximum is a sell signal). The profit or loss is then calculated with true gold prices using the signals. In addition, since there are trading platforms with commission free trades, a 0% buy and sell commission will be assumed for the profit/loss calculations.

The baseline model with no external regressors predicted one week in advance with a RMSE of 76.34. According to the baseline forecast plot in Figure B5, the predicted values did not follow the same price trend as gold. Due to only a one week forecast, the profit/loss will not be calculated since it would only have made one trade during that time period. The second model included 5 day lagged external regressors with a weekly rollover partitioning for the forecast. The model predicts one week at a time for approximately 1 month which resulted in an RMSE of 58.61. According to plot Figure B8, the forecast followed closely with the true values for the first half of the month, but significantly deviated from each other in the last week of the prediction when there was a rapid increase in gold price. With respect to trading, the forecasted model would have made 6 trades with a profit of 4.43%.

The third model included 1 day lagged external regressors with a daily rollover partitioning for the forecast. The model predicts one day in advance for approximately 1 month and has a calculated RMSE score of 43.45. Moreover, the predicted values followed the general

trend as the true values when comparing the predicted versus true values in Figure B11. In other words, the forecast model seemed to be accurate when predicting both the downward and upward trend within the 1 month timeframe. With regards to profit/loss, the model would have made 4 trades with a profit of 4.27%.

The last model also incorporates a daily rollover partitioning for the forecast. However, the purpose of this model is to demonstrate forecast ability with true external regressor data. In other words, the external regressor data will have no lag in order to illustrate if gold forecasting is possible. The result is an RMSE of 39.46 with a trend very similar to the true values as shown in Figure B14. However, with perfect external regressor data, the forecast was still not able to predict the sudden rapid increase of gold's price due to gold's random walk characteristics. Furthermore, the model would have made 5 trades for a profit of 6.10%.

In terms of RMSE, the model with a daily rollover partitioning had the lowest value of 43.45 compared to the weekly rollover partitioning at 58.61. The RMSE of the model that used true external regressor data had an RMSE of 39.46 which suggests that forecasting the external regressor data accurately leads to better forecasting performance than using lagged external data. In addition, the model with external regressors validates the assumption that if US Money Supply, DOW Jones Industrial Average, Google Trend Hits, Federal Funds Data, and Silver prices were used as external regressors, then the general trend of gold can be forecasted. Moreover, all models also returned a profit of between 4-6%, indicating that the trading strategy by selling and buying at local maximums and minimums of the forecasted data could work for investors. However, the models would have not had a higher profit percentage if gold was held for the duration of the month; roughly a 7% increase in profit. Although, one advantage of

forecasting and trading a week at a time is that it is safer than holding an asset for the unseeable future depending on the asset.

Moving Average

A simple moving average (SMA) was used to show the current price trends and the potential for a change in trend of gold prices (in USD). This method can be defined as "an arithmetic moving average calculated by adding the recent prices and then dividing that value by the number of time periods in the calculation average" (RPubs). The function and package to visualize the trend of the gold prices was quantmod and can be observed in Figure B16. This method was to simply observe any trends that exist with the prices of gold over time and was not able to forecast or combine any external predictors, thus would be suggested to only supplement the investors decision making process.

Conclusion

Future works that plan on using the Prophet model on a daily or weekly basis should include an API or a data pipeline to feed in data for the target variable and external regressors to ensure up-to-date information without the need to redownload the data manually. In addition, forecasting the external data can lead to better forecasting performance rather than using lagged data. As well as, to include different external regressors that are correlated with gold to help forecast the prices more accurately. In regards to the Prophet model, there are a vast amount of parameters to tune for this model. Therefore, better parameter tuning would also be suggested to increase its accuracy. Additionally, different time periods should also be tested for the forecasting trading strategy. One potential reason for why all the models returning a profit instead of a loss is due to the recent price increase of gold.

In conclusion, the results observed for each of the selected models in an attempt to predict the prices of gold led to poor prediction accuracy; the team concluded that it was suggested that the Prophet model would be beneficial for investors to incorporate the model rather for decision making processes to observe a general trend of the gold prices. The logistic regression models would not be recommended as the volatility associated was of high concern and did not meet the selected performance criteria of accuracy, RMSE, trend to gold prices, and the calculated profits. The moving average model was only suggested to be a supplement for the decision making process as well as it provided a clear visual representation of the trends that existed in gold prices.

References

Chan Phooi M'ng, J., & Zainudin, R. (2016). *Assessing the Efficacy of Adjustable Moving Averages Using ASEAN-5 Currencies*. PLOS ONE, 11(8), e0160931. https://doi.org/10.1371/journal.pone.0160931

Chirwa, T. G., & Odhiambo, N. M. (2020). *Determinants of gold price movements: An empirical investigation in the presence of multiple structural breaks*. Resources Policy, 69, 101818. https://doi.org/10.1016/j.resourpol.2020.101818

Jones, C. I. (2021). *Macroeconomics*. W.W. Norton & Company.

Omar, A. B., Huang, S., Salameh, A. A., Khurram, H., & Fareed, M. (2022). Stock Market Forecasting Using the Random Forest and Deep Neural Network Models Before and During the COVID-19 Period. Frontiers in Environmental Science, 10. https://doi.org/10.3389/fenvs.2022.917047

RPubs - Simple Moving Average. (n.d.). Rpubs.com. https://rpubs.com/techanswers88/SimpleMovingAverage

Ruiz-Cruz, R. (2018). *Portfolio modeling for an algorithmic trading based on control theory*. IFAC-PapersOnLine, 51(13), 390–395. https://doi.org/10.1016/j.ifacol.2018.07.310

Shmueli, G. & Lichtendahl Jr., K.C. (2018). *Practical time series forecasting with R: A hands-on guide (2nd ed.)*. Axelrod Schnall Publishers.

Appendix A

Table A1

Dataset Files and Description

Filename	Description
Gold.csv	Description: Includes the real time gold prices(in USD). Soure: Nasdaq
Gold_Trend_hits.csv	Description: Includes the total hits, which are a method of monitoring the traffic on a specific website related to gold. Soure: Google
UNRATE.csv	Description: Unemployment Rate Soure: Federal Reserve Economic Database
US Dollar Index Historical Data.csv	Description: Is an index (or measure) of the value of the United States dollar relative to a basket of foreign currencies. Soure: Wall Street Journal
WM2NS.csv	Description: M2 is a measure of the money supply that includes cash, checking deposits, and easily-convertible near money. Soure: Federal Reserve Economic Database
DJIA.csv	Description: Dow Jones Industrial Average Soure: Federal Reserve Economic Database
DFF.csv	Description: Federal Funds Rate Soure: Federal Reserve Economic Database
Silver	Description: Includes the real time silver prices (in USD). Soure: Quandl

Table A2

Confusion Matrix for Logistic Regression Model

	Actual = 0	Actual = 1
Prediciton = 0	2	4
Prediction = 1	1	9

Appendix B

Figure B1

Correlation Heatmap

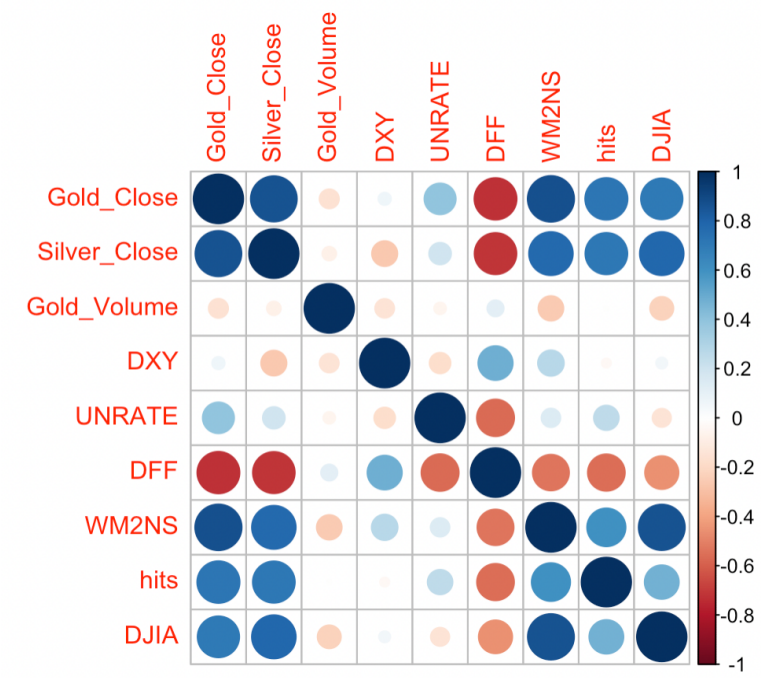


Figure B2

Time Series Trend of Normalized Variables

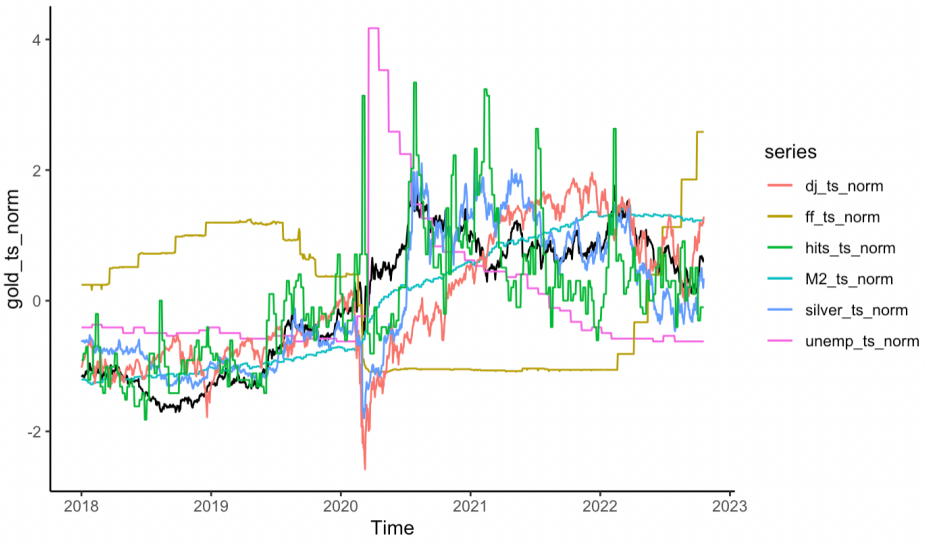


Figure B3

Scatterplot of M2 and Gold Close price with Date Overlay

Figure B4

Prophet Forecast Plot for Gold: Baseline Prediction

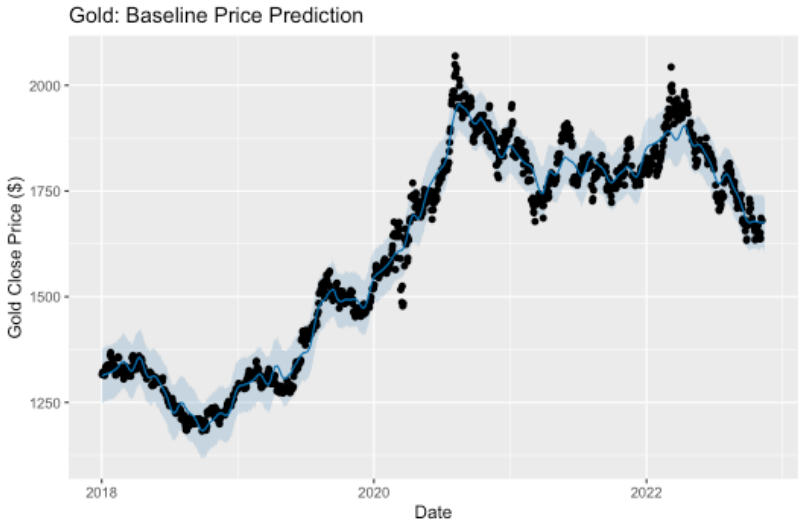


Figure B5

Predicted(blue) Versus Actual(red) for a 1 Week Forecast of the Baseline Forecast Model

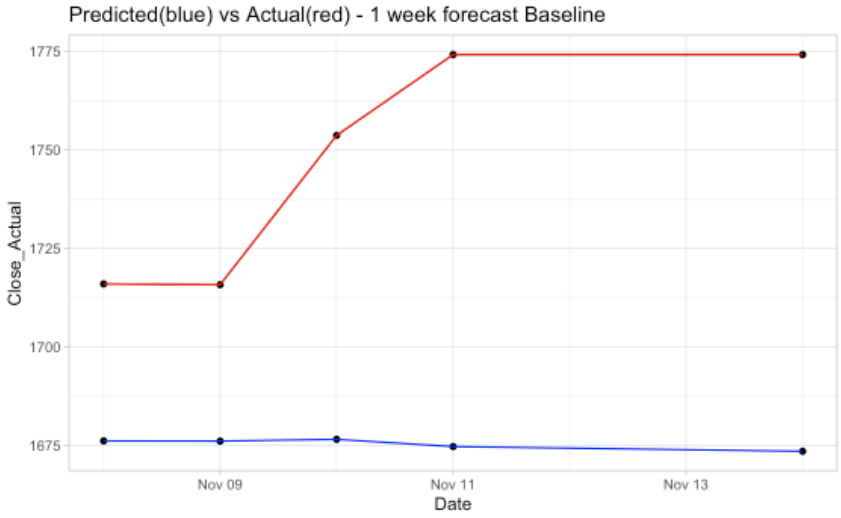


Figure B6

Time Series Decomposition of the Baseline Forecast Model

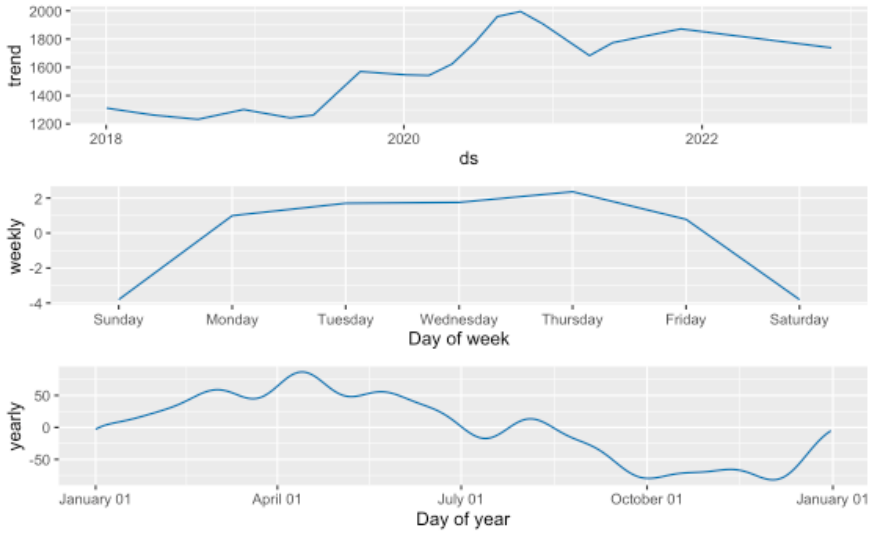


Figure B7

Prophet Forecast Plot for Gold 1 Week Rollover Partitioning

Figure B8

Predicted(blue) Versus Actual(red) for a One Month Gold Forecast Using One Week Rollover Partitioning

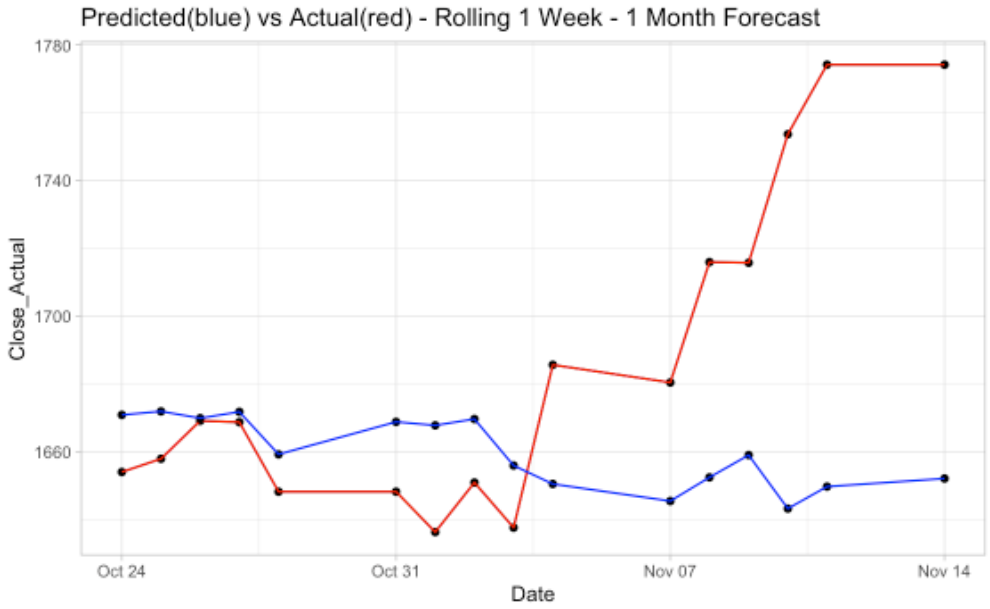


Figure B9

Time Series Decomposition of the 1 Week Rollover Gold Forecast Model

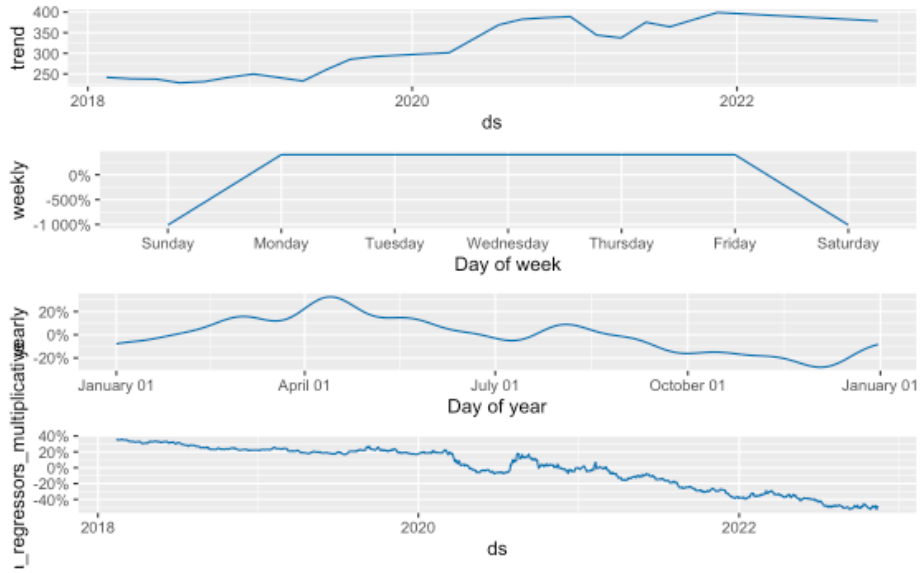


Figure B10

Prophet Forecast Plot for Gold 1 Day Rollover Partitioning

Figure B11

Predicted(blue) Versus Actual(red) for a One Month Gold Forecast Using One Day Rollover Partitioning

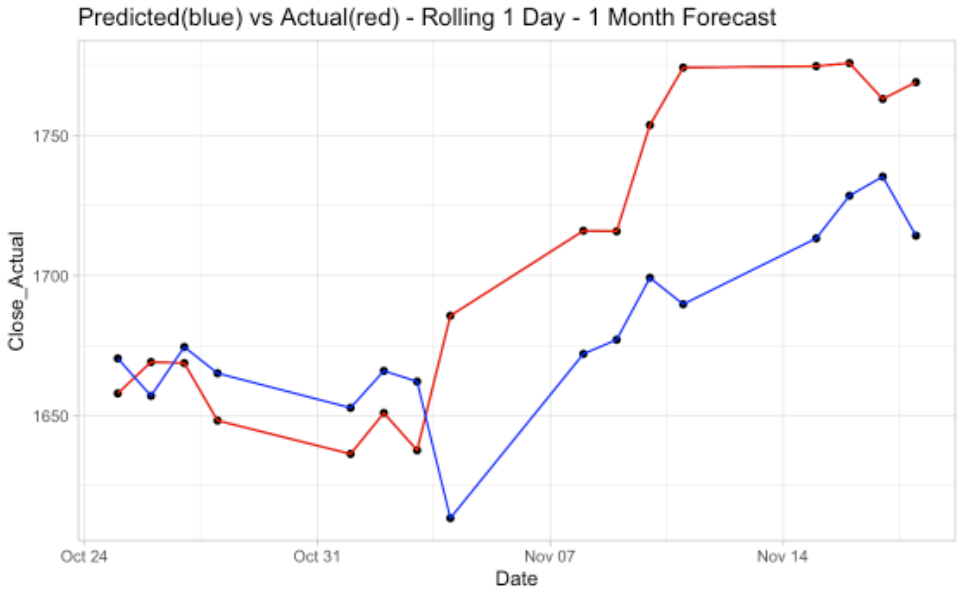


Figure B12

Time Series Decomposition of the 1 Day Rollover Gold Forecast Model

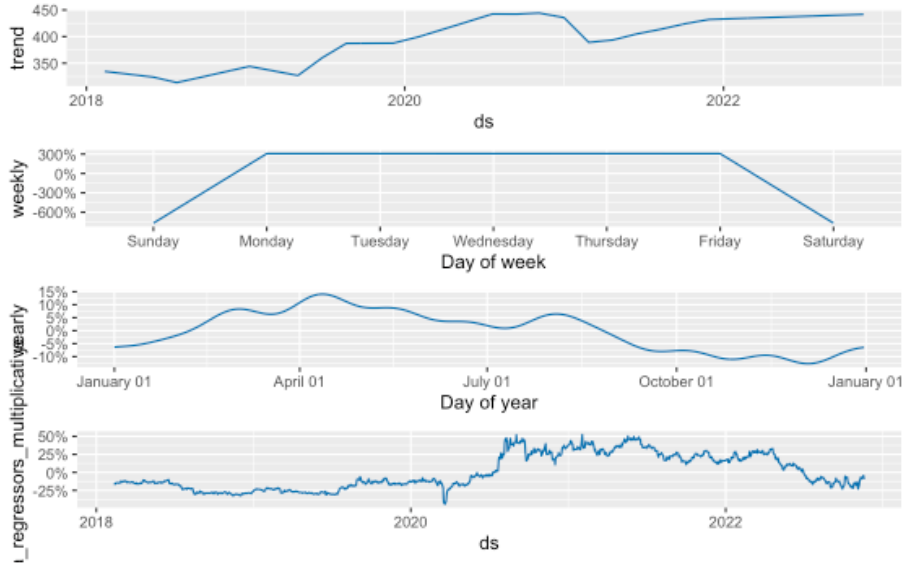


Figure B13

Prophet Forecast Plot for Gold 1 Day Rollover Partitioning with No Lagged Predictors

Figure B14

Predicted(blue) Versus Actual(red) for a One Month Gold Forecast Using One Day Rollover Partitioning with No Lagged Predictors

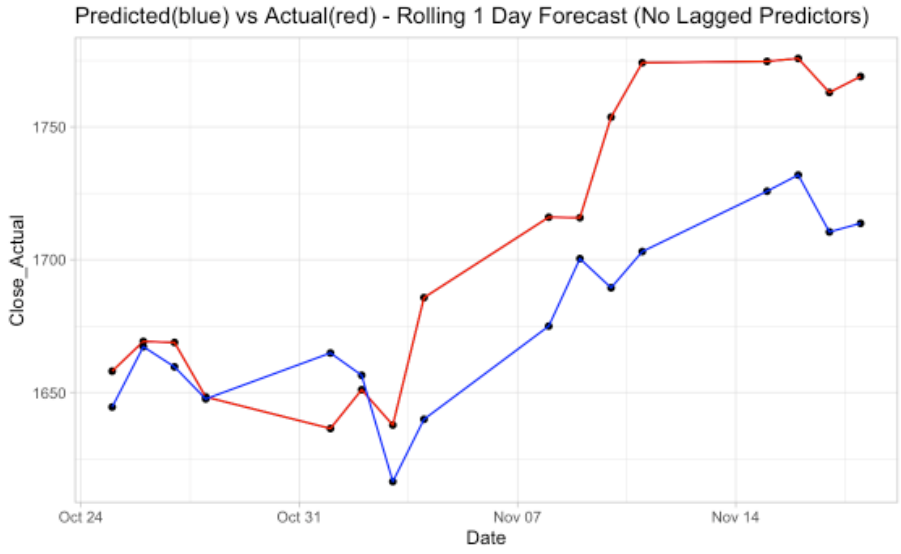


Figure B15

Time Series Decomposition of the 1 Day Rollover Gold Forecast Model with no Lagged Predictors

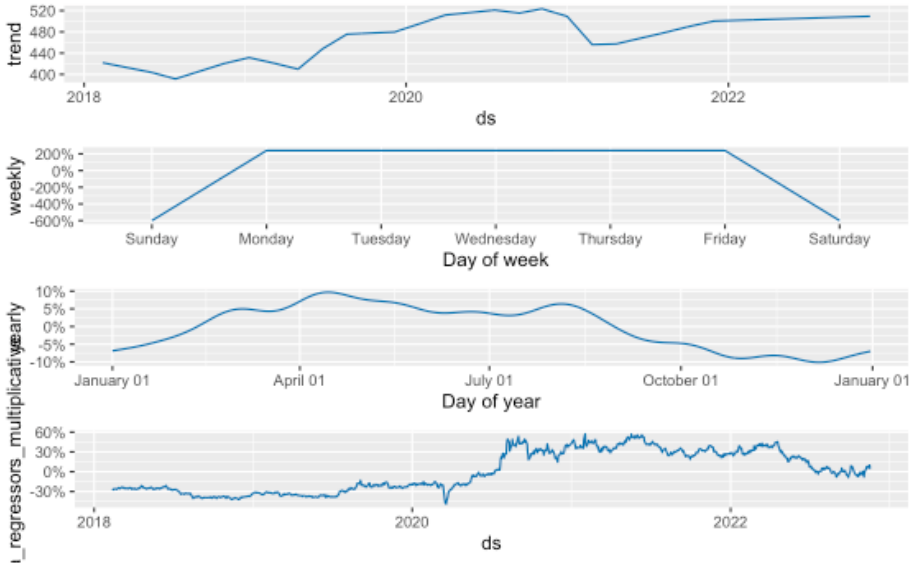


Figure B16

Simple Moving Average of Gold Prices

Gold Forecast Prophet

Amin, Kyle, Ryan

11/24/2022

Load Data

```
# Load Data

gold <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Series Pro
ject/Update/Gold.csv",
    col_types = cols(Date = col_date(format = "%m/%d/%Y")))

unemp <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Series Pr
oject/Update/UNRATE.csv", show_col_types = FALSE)

M2 <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Series Proje
ct/Update/WM2NS.csv", show_col_types = FALSE)

DJ <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Series Proje
ct/Update/DJIA.csv",  show_col_types = FALSE)

fed_funds <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Serie
s Project/Update/DFF.csv", show_col_types = FALSE)

silver <- Quandl('LBMA/SILVER')

dollar_index <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Se
ries Project/Update/US Dollar Index Historical Data.csv", col_types = cols(Date = col
_date(format = "%m/%d/%Y")))

Google <- read_csv("/Users/datascience/Desktop/Time Series Data Science/Time Series P
roject/Update/Gold_Trend_hits.csv", show_col_types = FALSE)
```

Create Candlestick/volume graph of gold

```
#edit the column names in data frames
Price_plot <- gold %>%
  plot_ly(x = ~Date,
          type = "candlestick",
          open = ~Open,
          close = ~`Close/Last`,
          high = ~High,
          low = ~Low,
          name = "price") %>%
```

```
  layout(
    xaxis = list(
      rangeselector = list(
        buttons = list(
          list(
            count = 3,
            label = "3 mo",
            step = "month",
            stepmode = "backward"),
          list(
            count = 6,
            label = "6 mo",
            step = "month",
            stepmode = "backward"),
          list(
            count = 1,
            label = "1 yr",
            step = "year",
            stepmode = "backward"),
          list(
            count = 2,
            label = "2 yr",
            step = "year",
            stepmode = "backward"),
          list(
            count = 3,
            label = "3 yr",
            step = "year",
            stepmode = "backward"),
          list(
            count = 5,
            label = "5 yr",
            step = "year",
            stepmode = "backward"),
          list(step = "all"))),

      rangeslider = list(visible = FALSE)),
         yaxis = list(title = "Price ($)",
                      showgrid = TRUE,
                      showticklabels = TRUE))
Volume <- select(gold, Date, Volume)
Volume$Date <- as.Date(Volume$Date , format = "%m/%d/%y")
Volume$Vol <- as.numeric(as.character(Volume$Volume)) / 1000
Volume_plot <- Volume %>%
  plot_ly(x=~Date, y=~Vol, type='bar', name = "Volume") %>%
  layout(yaxis = list(title = "Volume (Units of Thousand)"))
plot <- subplot(Price_plot, Volume_plot, heights = c(0.7,0.3), nrows=2,
             shareX = TRUE, titleY = TRUE) %>%
  layout(title = 'GC:CMX')
plot
```

Adjust data types where needed

```
#change data type from chr to date in the gold data set
gold$Date <- as.Date(gold$Date, format = "%m/%d/%y")
gold$DATE <- gold$Date
#change the datatype to date in the DJIA data set
DJ$DATE <- as.Date(DJ$DATE, format = "%m/%d/%y")
#indicate columns to keep in gold data set
keep_cols <-  c("Date","Close/Last", 'Volume')
gold <- gold[keep_cols]
gold <- gold %>%
  rename(Gold_Close = 'Close/Last', Gold_Volume ='Volume')
```

Create the daily M2 rate data frame (in billions)

```
#create all dates from the date column in the data frame
all_dates <- M2 %>%select(DATE) %>%
  complete(DATE=  seq.Date(min(DATE), max(DATE), by="day"),
           ) %>% mutate(TIME=paste(year(DATE),str_pad(month(DATE), 2, pad = "0"),sep
= "-"))
#join the all_dates object back onto original datafarme
M2_daily <- left_join(all_dates,M2,by="DATE")
#fill the NA values from the join with the first value from the dataset
M2_daily <- na_locf(M2_daily, option = "locf")
keep_cols <- c("DATE", "WM2NS")
M2_daily <- M2_daily[keep_cols]
M2_daily <- M2_daily %>%
  select(c("DATE","WM2NS")) %>%
  rename(Date = DATE)
```

Create the daily unemployment rate data frame

```
#create all dates from the date column in the dataframe
all_dates <- unemp %>%select(DATE) %>%
  complete(DATE=  seq.Date(min(DATE), max(DATE), by="day"),
           ) %>% mutate(TIME=paste(year(DATE),str_pad(month(DATE), 2, pad = "0"),sep
= "-"))
#join the all_dates object back onto original datafarme
unemp_daily <- left_join(all_dates,unemp,by="DATE")
#fill the NA values from the join with the first value from the dataset
unemp_daily <- na_locf(unemp_daily,  option = "locf")
keep_cols <- c("DATE", "UNRATE")
unemp_daily <- unemp_daily[keep_cols]
unemp_daily <- unemp_daily %>%
  select(c("DATE","UNRATE")) %>%
  rename(Date = DATE)
```

Create the daily federal funds data frame

```
#create all dates from the date column in the data frame
all_dates <- fed_funds %>%select(observation_date) %>%
  complete(observation_date =  seq.Date(min(observation_date), max(observation_date),
by="day"),
           ) %>% mutate(TIME=paste(year(observation_date),str_pad(month(observation_d
ate), 2, pad = "0"),sep = "-"))
#join the all_dates object back onto original dataframe
ff_daily <- left_join(all_dates,fed_funds,by="observation_date")
#fill the NA values from the join with the first value from the dataset
ff_daily <- na_locf(ff_daily, option = "locf")
keep_cols <- c("observation_date", "DFF")
ff_daily <- ff_daily[keep_cols]
ff_daily <- ff_daily %>%
  select(c("observation_date","DFF")) %>%
  rename(Date = observation_date)
```

Create the daily Dow Jones IA data frame

```
#create all dates from the date column in the data frame
all_dates <- DJ %>%select(DATE) %>%
  complete(DATE =  seq.Date(min(DATE), max(DATE), by="day"),
           ) %>% mutate(TIME=paste(year(DATE),str_pad(month(DATE), 2, pad = "0"),sep
= "-"))
#join the all_dates object back onto original datafarme
dj_daily <- left_join(all_dates, DJ ,by="DATE")

#fill the NA values from the join with the first value from the dataset
dj_daily <- na_locf(dj_daily,  option = "locf")
keep_cols <- c("DATE", "DJIA")
dj_daily <- dj_daily[keep_cols]
dj_daily <- dj_daily %>%
  select(c("DATE", "DJIA")) %>%
  rename(Date = DATE)
```

Create the daily Silver Price data frame

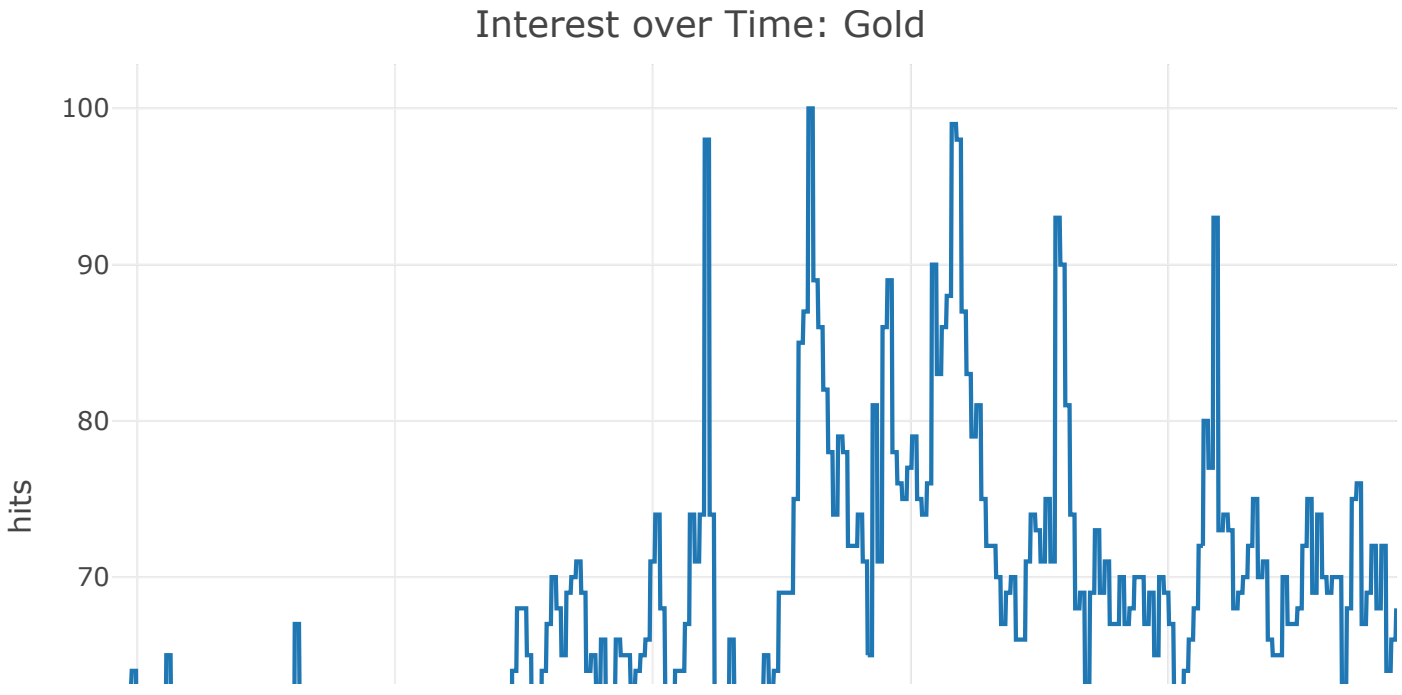
```
# add in Silver
silver <- silver[,c('Date', 'USD')]
silver <- silver %>%
  rename(Silver_Close = 'USD')
# Join Silver and Gold Data
Metals <- inner_join(silver, gold, by="Date")
```

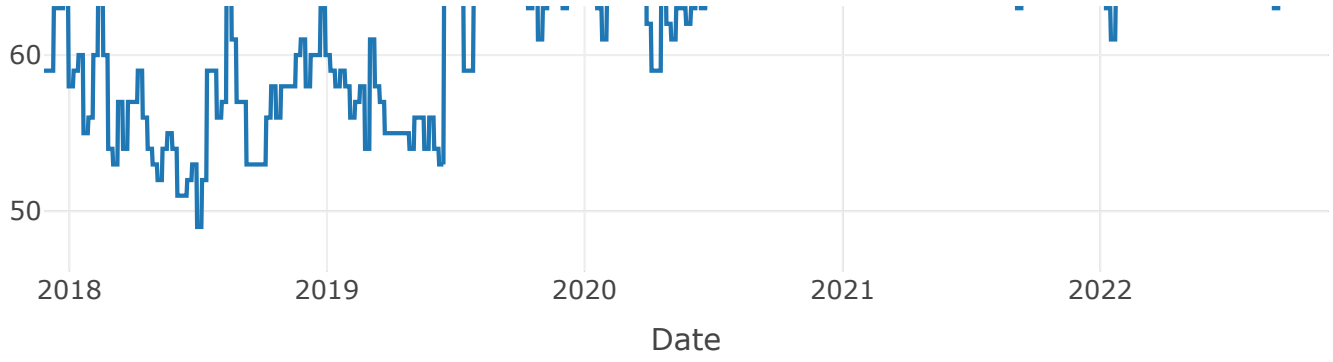
Create the Dollar Index Price data frame

```
# Add in Dollar Index
dollar_index <- dollar_index %>%
  select(c("Date","Price")) %>%
  rename(DXY = Price)
```

Google trends to find hits on Gold worlwide

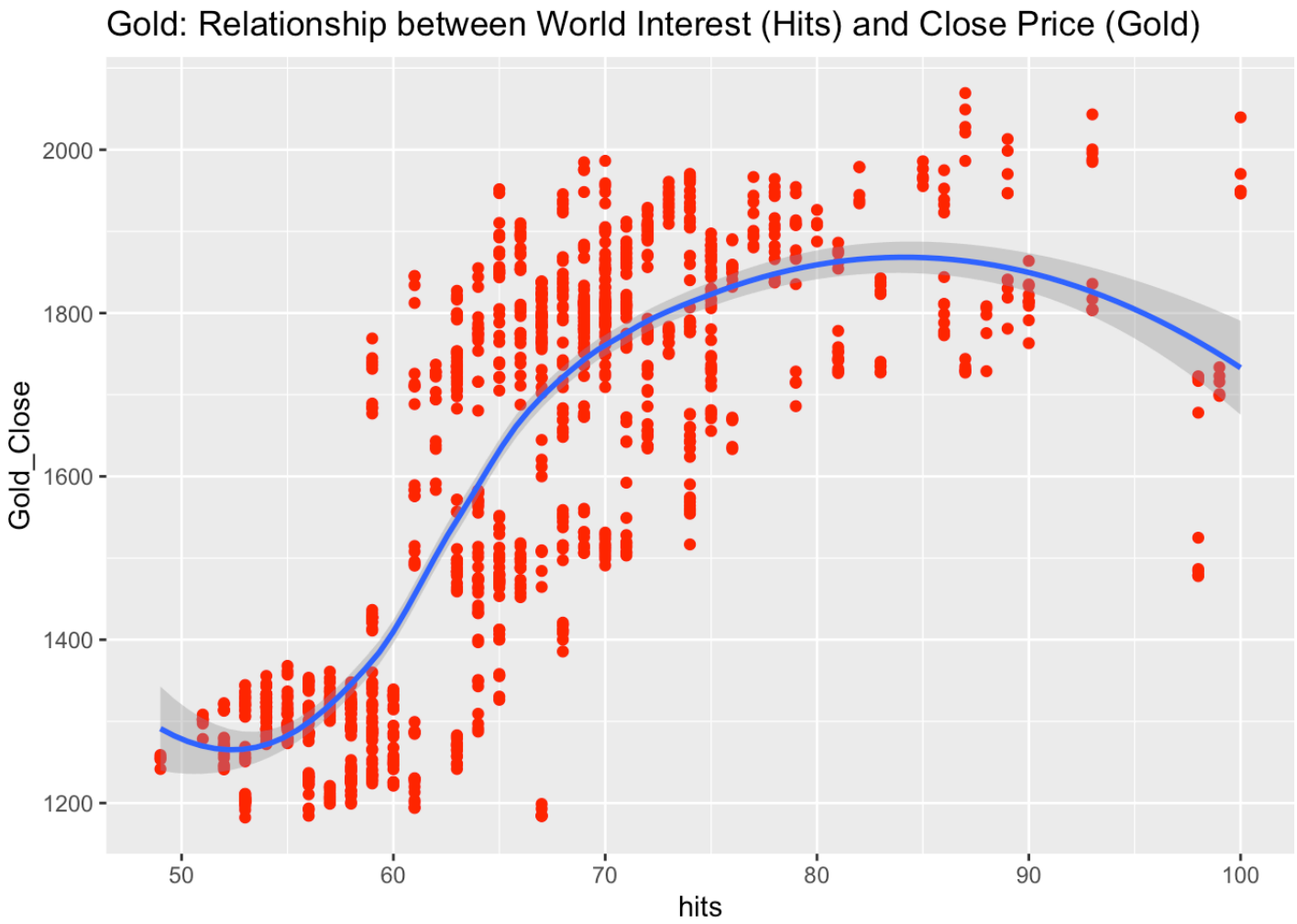
```
#trends <- gtrends(keyword = "Gold", onlyInterest = TRUE, time = "today+5-y")

#create all dates from the date column in the data frame
all_dates <- Google %>%select(Date) %>%
  complete(Date =  seq.Date(min(Date), max(Date), by="day"),
           ) %>% mutate(TIME=paste(year(Date),str_pad(month(Date), 2, pad = "0"),sep
= "-"))
#join the all_dates object back onto original datafarme
Google_daily <- left_join(all_dates, Google ,by="Date")
#fill the NA values from the join with the first value from the dataset
Google_daily <- na_locf(Google_daily,  option = "locf")

# Create Daily Data Fame
keep_cols <- c("Date", "Gold: (Worldwide)")
Google_daily <- Google_daily[keep_cols]
Google_daily <- Google_daily %>%
  select(c("Date", "Gold: (Worldwide)")) %>%
  rename(hits = 'Gold: (Worldwide)')
Google_daily %>%
  plot_ly(type='scatter',x=~Date, y=~hits, mode = 'lines', name = "Google Search Tren
ds") %>%
  layout(title = paste0("Interest over Time: ", "Gold"), yaxis = list(title = "hits")
)
```






```
trends_daily <- Google_daily
# Hits verse Gold
trends_daily %>%
  left_join(gold, by = "Date") %>%
  select(one_of(c("Date", "hits", "Gold_Close"))) %>%
  drop_na() %>%
  ggplot(aes(hits, Gold_Close)) + geom_point(color="red") + geom_smooth(method = 'loe
ss') +
  labs(title =paste0("Gold",": Relationship between World Interest (Hits) and Close P
rice (Gold)"))
```

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



Merge data frames and create the final dataframe for EDA

```
#merge the dataframes together on dates
full_df <- left_join(Metals, dollar_index, by="Date")
full_df <- left_join(full_df, unemp_daily, by="Date")
full_df <- left_join(full_df, ff_daily, by="Date")
full_df <- left_join(full_df, M2_daily, by="Date")
full_df <- left_join(full_df, Google_daily, by="Date")
full_df <- left_join(full_df, dj_daily, by="Date")
full_df <- left_join(full_df, trends_daily, by="Date")
full_df$Gold_Volume <- na_ma(full_df$Gold_Volume, k=1, weighting = "simple")
keep_cols <- c("Date", "Silver_Close", "Gold_Close","Gold_Volume", "DXY", "UNRATE", "
DFF", "WM2NS" ,"hits.x",     "DJIA")
full_df <- full_df[keep_cols]
full_df <- full_df %>%
  rename(hits = hits.x)
```

```
# Check for any NA Values
cbind(
   lapply(
     lapply(full_df, is.na)
     , sum)
   )
```

```
##              [,1]
## Date         0
## Silver_Close 0
## Gold_Close   0
## Gold_Volume  0
## DXY          0
## UNRATE       1255
## DFF          1255
## WM2NS        1259
## hits         1259
## DJIA         1255
```

Subset the data frame with values from 2018 forward

```
full_df <- full_df %>% arrange(ymd(full_df$Date))
full_df <- full_df[full_df$Date >= "2018-01-01",]
#Replace NA with the most recent value
full_df$WM2NS <- zoo::na.fill(full_df$WM2NS, "extend")
full_df$hits <- zoo::na.fill(full_df$hits, "extend")
full_df$UNRATE <- zoo::na.fill(full_df$UNRATE, "extend")
tail(full_df)
```

	Date	Silver_Close	Gold_Close	Gold_Volume	DXY	UN…	D…	WM2…	hits
	<date>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2485	2022-11-15	21.940	1774.7	276828	106.40	3.5	3.83	21346.7	66
2486	2022-11-16	21.950	1775.8	191870	106.28	3.5	3.83	21346.7	66
2487	2022-11-17	21.075	1763.0	164553	106.69	3.5	3.83	21346.7	66
2488	2022-11-18	21.095	1769.0	29695	106.97	3.5	3.83	21346.7	66
2489	2022-11-21	20.640	1754.6	64912	107.78	3.5	3.83	21346.7	66
2490	2022-11-22	21.270	1754.8	57764	107.15	3.5	3.83	21346.7	66

6 rows | 1-10 of 11 columns

```
# Check for any NA Values
cbind(
   lapply(
     lapply(full_df, is.na)
     , sum)
   )
```

```
##              [,1]
## Date         0
## Silver_Close 0
## Gold_Close   0
## Gold_Volume  0
## DXY          0
## UNRATE       0
## DFF          0
## WM2NS        0
## hits         0
## DJIA         0
```

Create the correlation matrix for EDA

```
corr_fields <- c("Gold_Close", "Silver_Close","Gold_Volume", "DXY", "UNRATE", "DFF",
"WM2NS" ,"hits", "DJIA")
full_df$Gold_Volume <- as.numeric(full_df$Gold_Volume)
```

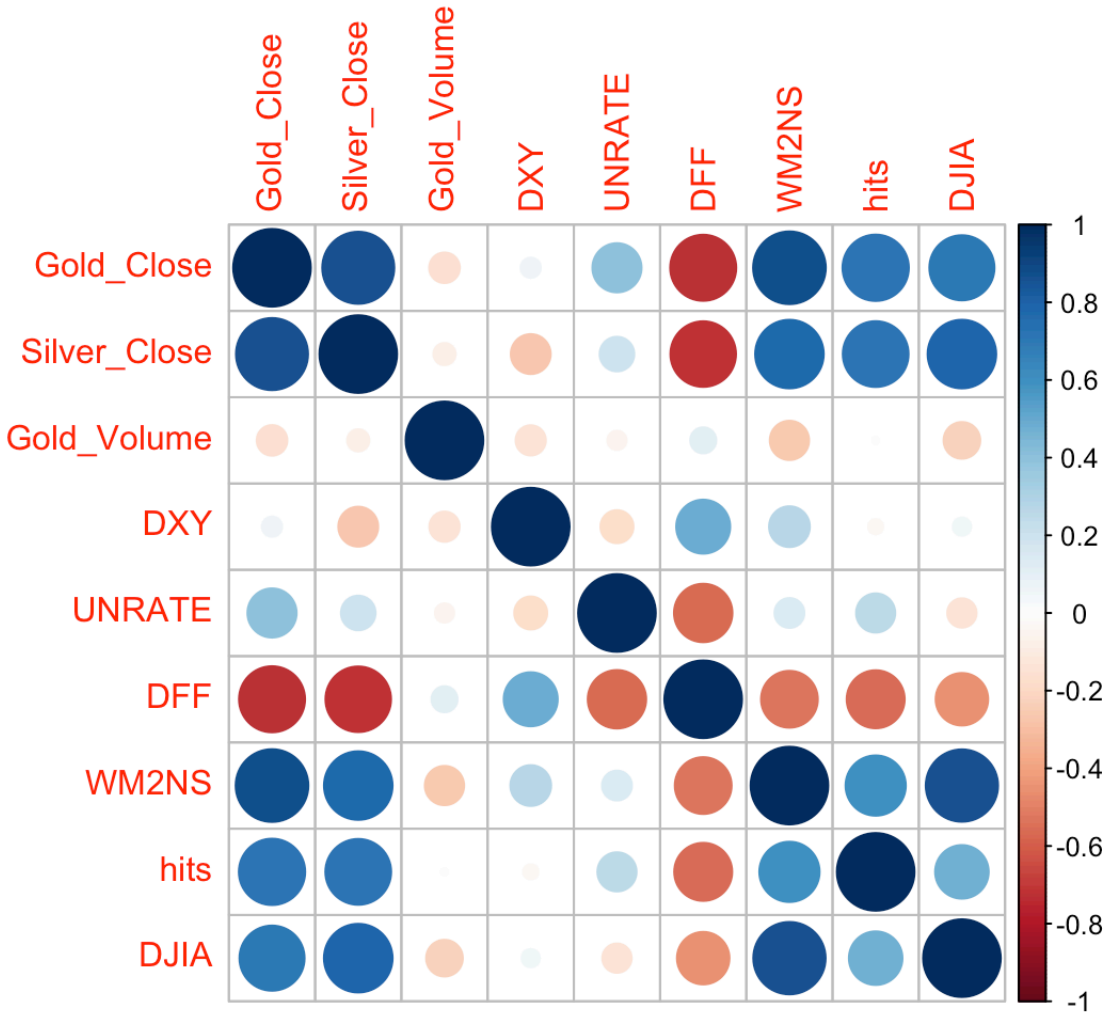
```
## Warning: NAs introduced by coercion
```

```
full_df$WM2NS <- as.numeric(full_df$WM2NS)
full_df$hits <- as.numeric(full_df$hits)

full_df$Gold_Volume <- na_ma(full_df$Gold_Volume, k=1, weighting = "simple")
full_df$WM2NS <- na_ma(full_df$WM2NS, k=1, weighting = "simple")
full_df$hits <- na_ma(full_df$hits, k=1, weighting = "simple")
```

```
corr_df <- full_df[corr_fields]
corr_matrix = cor(corr_df)
#display first row of correlation matrix
corr_matrix[,1]
```

```
##   Gold_Close Silver_Close  Gold_Volume          DXY       UNRATE          DFF
##   1.00000000   0.86040095  -0.15352950   0.06534308   0.39961962  -0.72279326
##        WM2NS         hits         DJIA
##   0.87544073   0.72864839   0.70205860
```

```
#develop the corrplot correlation matrix
corrplot(corr_matrix)
```

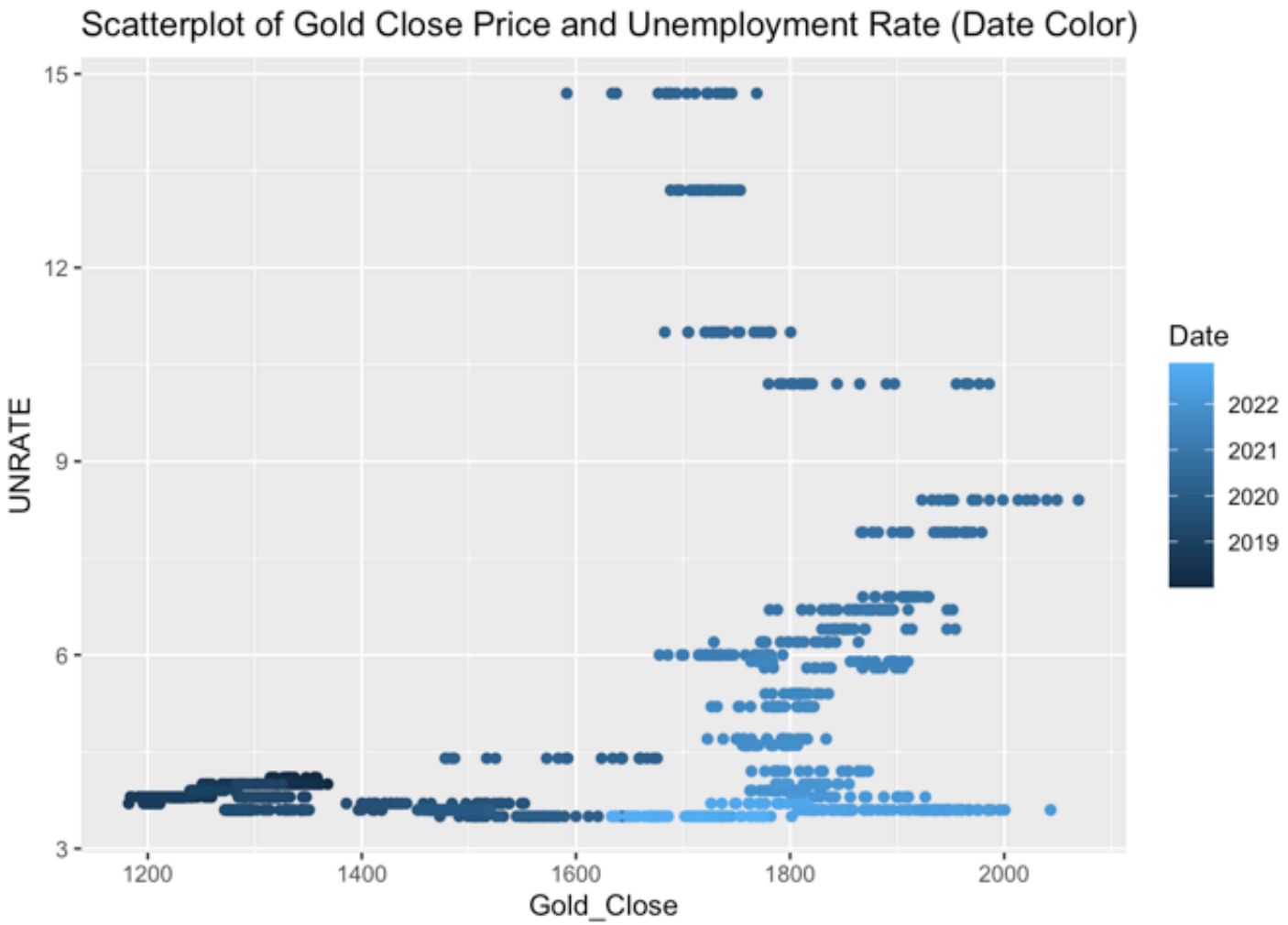


Silver Close, DFF, WM2NS, hits, and DIJA are strongly correlated with gold and therefore will be used as external regressors. ## Normalize variables for EDA

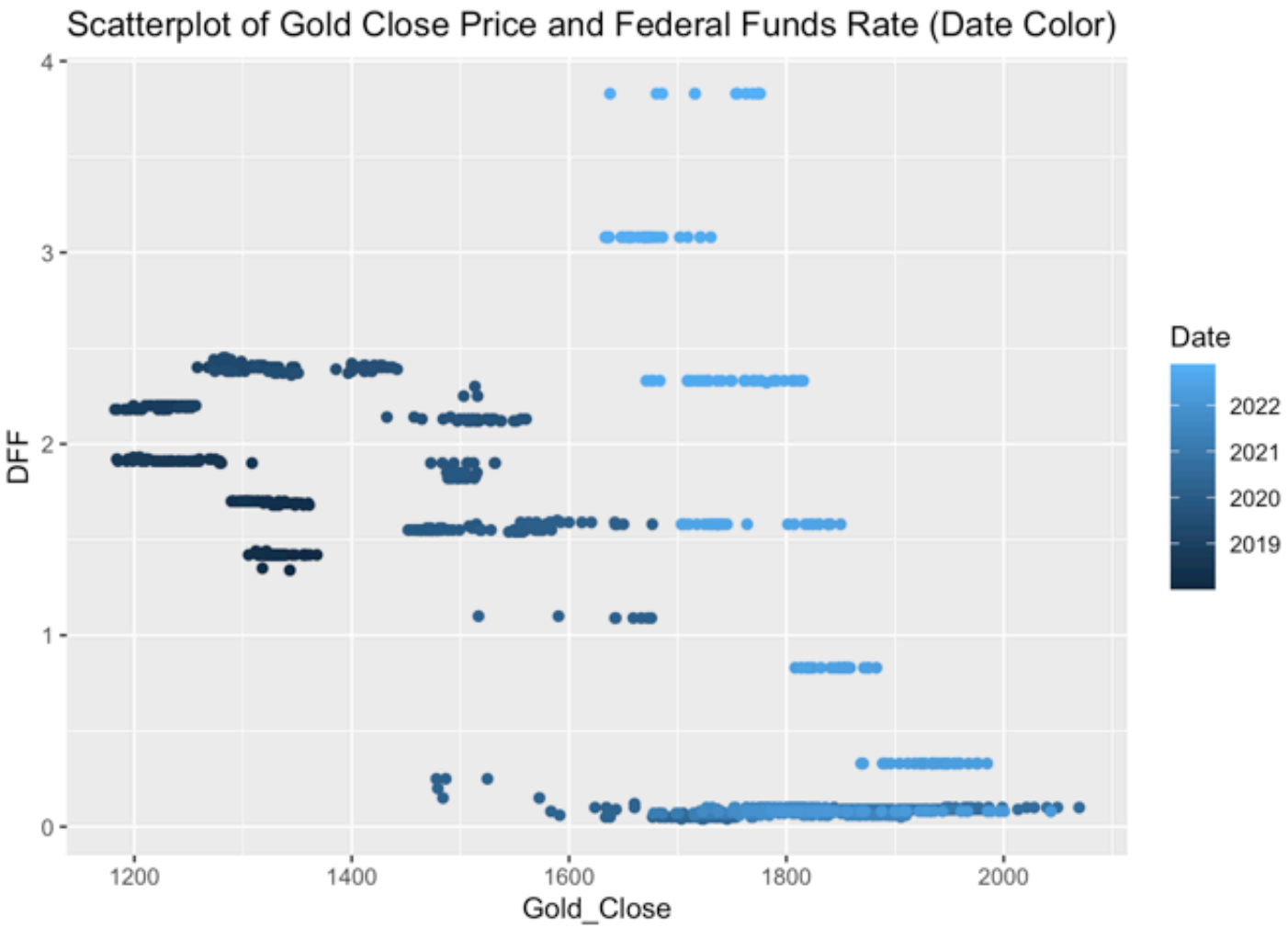
```
#normalize variables to view scaled relationships and add into full_df
full_df_Norm <- full_df
full_df_Norm[-1] <- lapply(full_df_Norm[-1], scale)
```

EDA with non-normalized varialbes

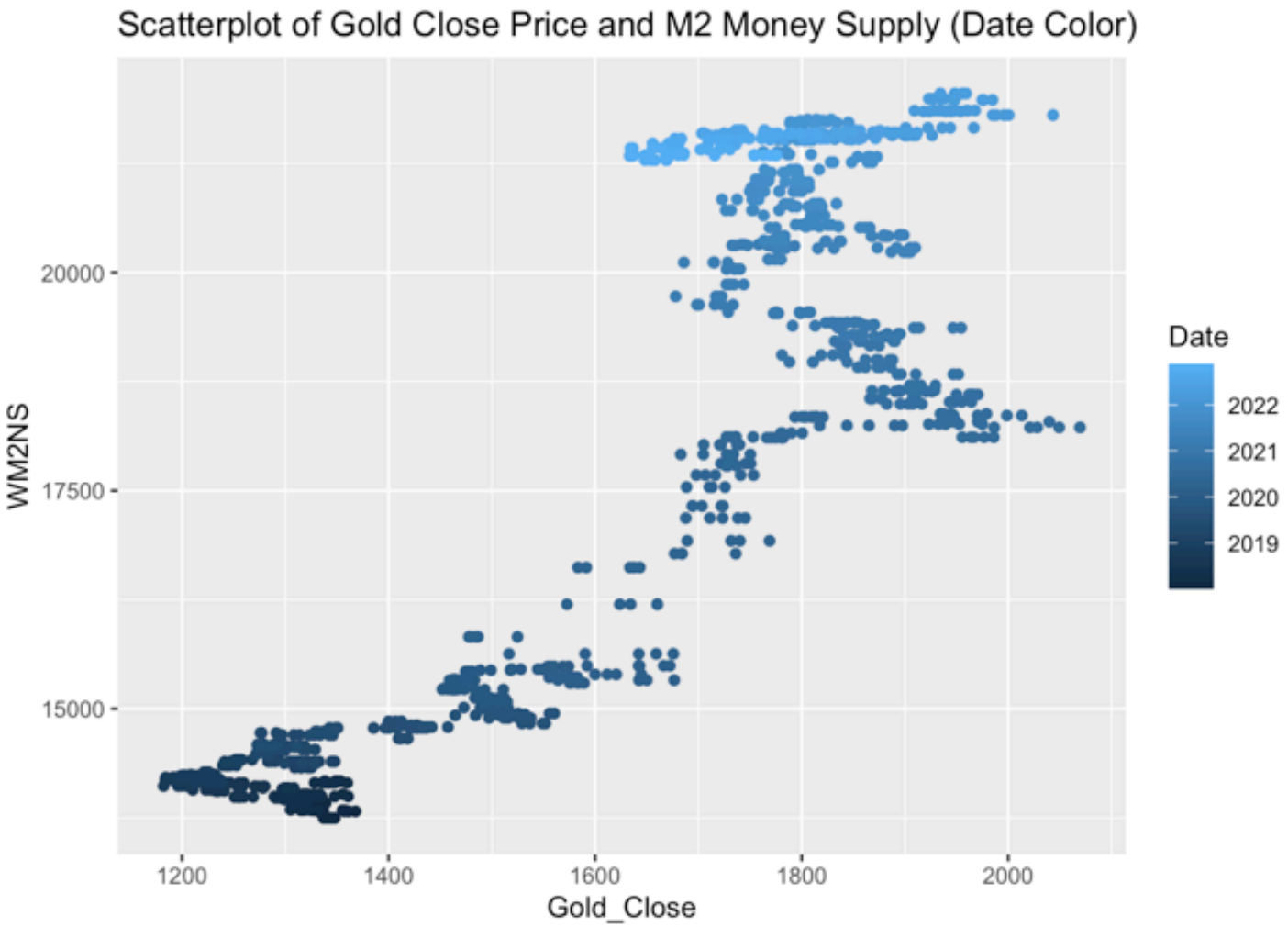
```
#scatterplot of gold & each external variable

ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = UNRATE, color = Date)) +
  ggtitle("Scatterplot of Gold Close Price and Unemployment Rate (Date Color)")
```



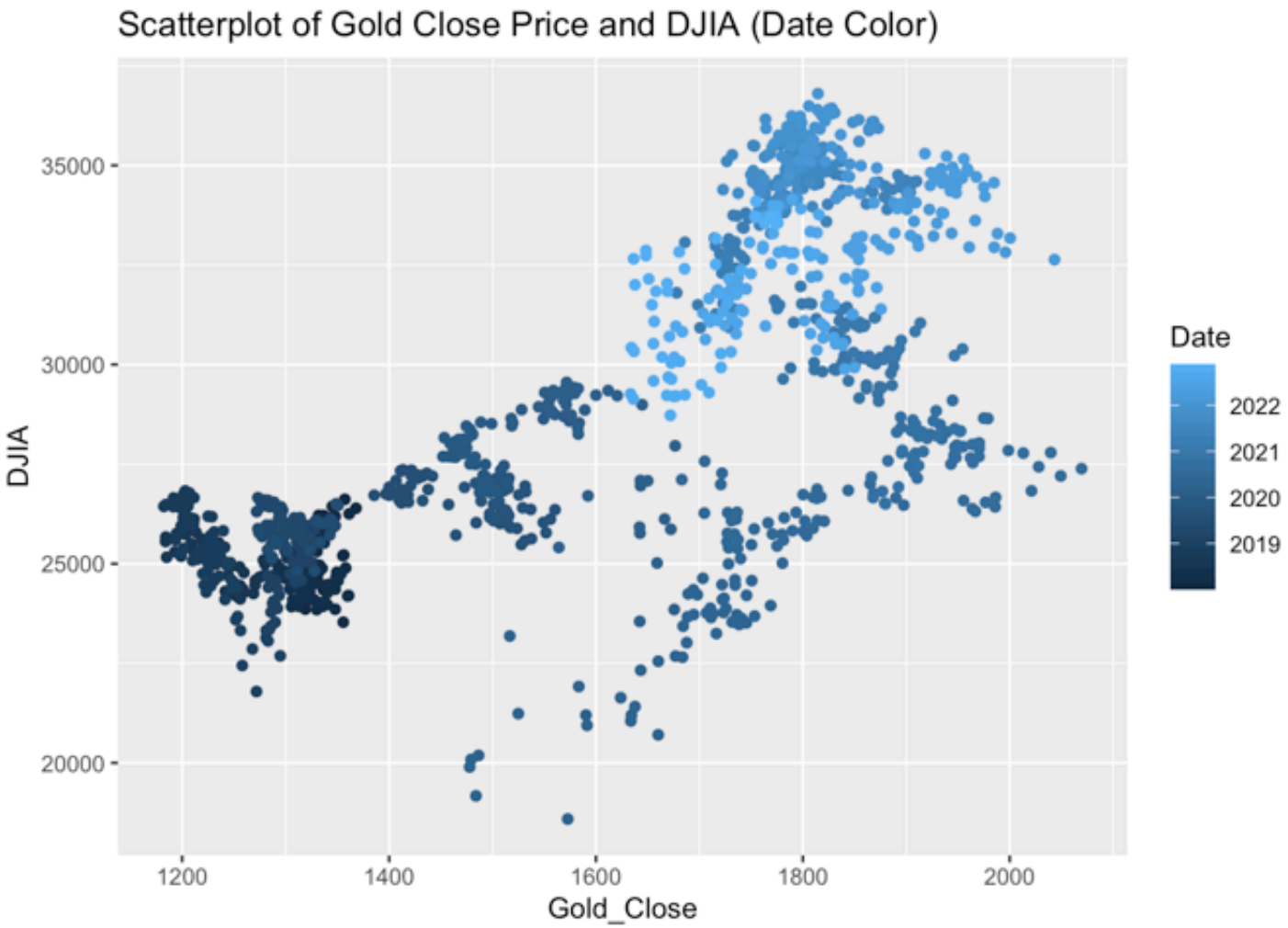

```
ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = DFF, color = Date)) +
  ggtitle("Scatterplot of Gold Close Price and Federal Funds Rate (Date Color)")
```



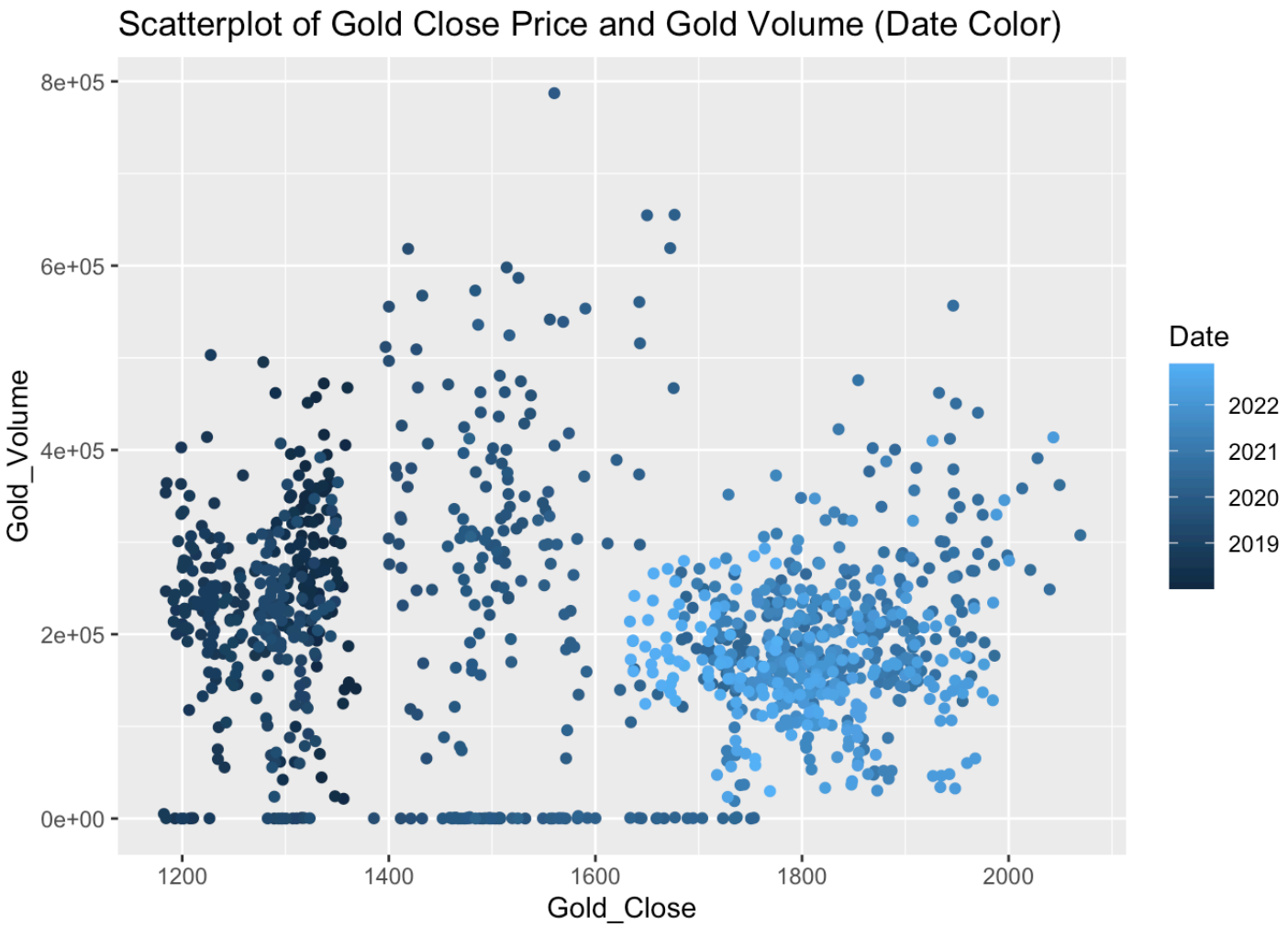

```
ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = WM2NS, color = Date)) +
  ggtitle("Scatterplot of Gold Close Price and M2 Money Supply (Date Color)")
```



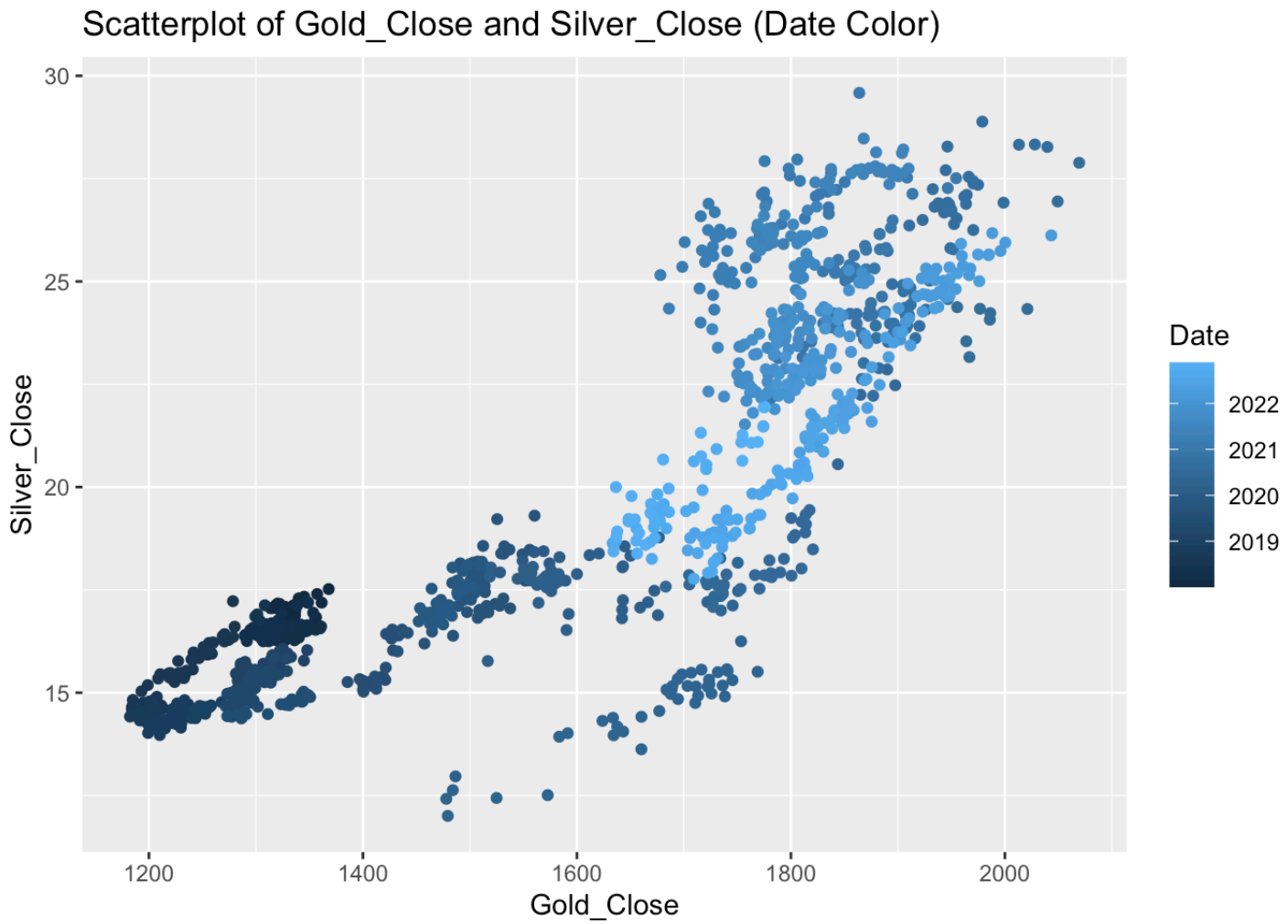

```
ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = DJIA, color = Date)) +
  ggtitle("Scatterplot of Gold Close Price and DJIA (Date Color)")
```



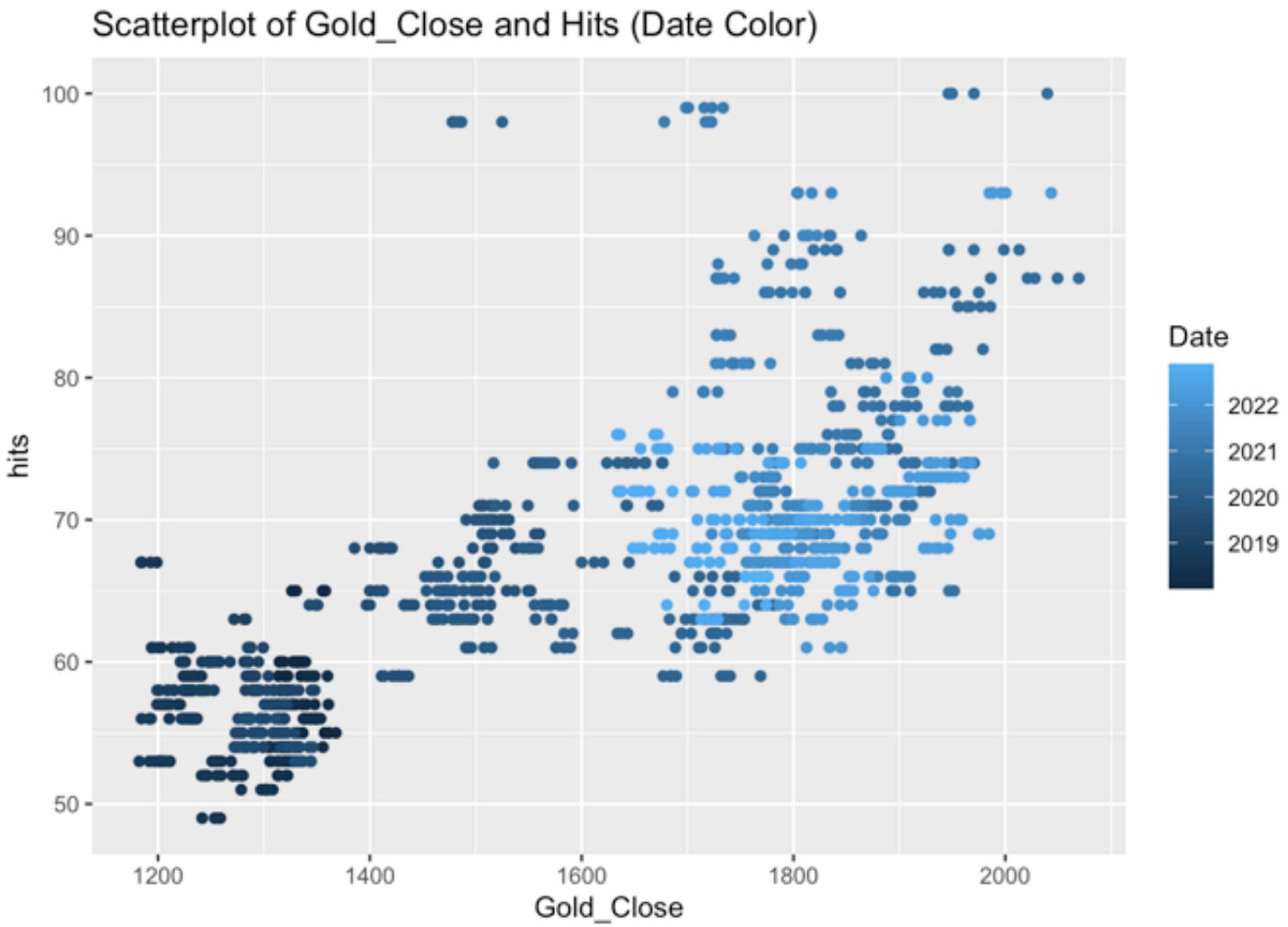

```
ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = Gold_Volume, color = Date)) +
  ggtitle("Scatterplot of Gold Close Price and Gold Volume (Date Color)")
```




```
ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = Silver_Close, color = Date)) +
  ggtitle("Scatterplot of Gold_Close and Silver_Close (Date Color)")
```




```
ggplot(data = full_df) +
  geom_point(mapping = aes(x = Gold_Close,
                           y = hits, color = Date)) +
  ggtitle("Scatterplot of Gold_Close and Hits (Date Color)")
```

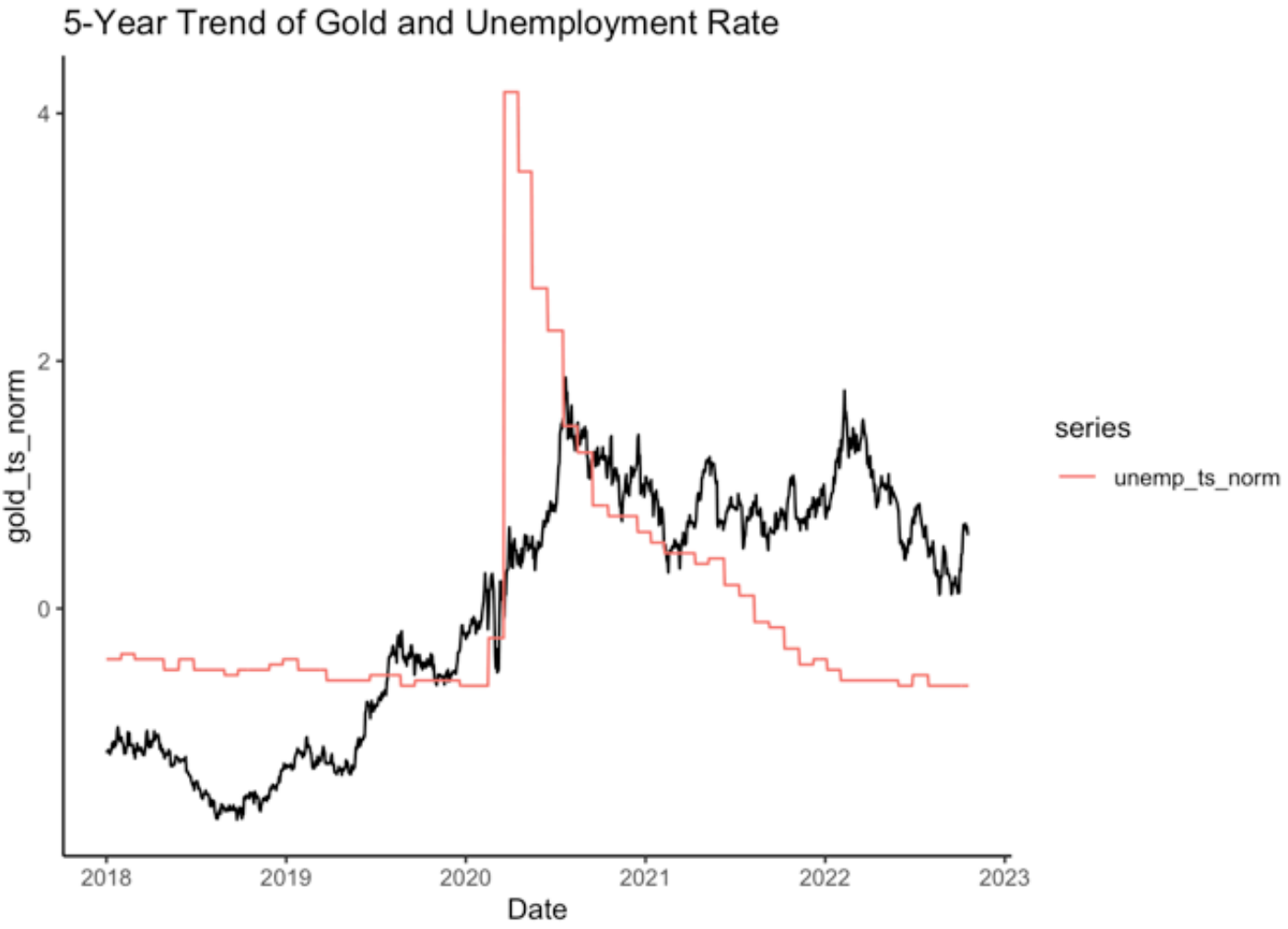


Create dataframe for Normalize EDA (on same axis)

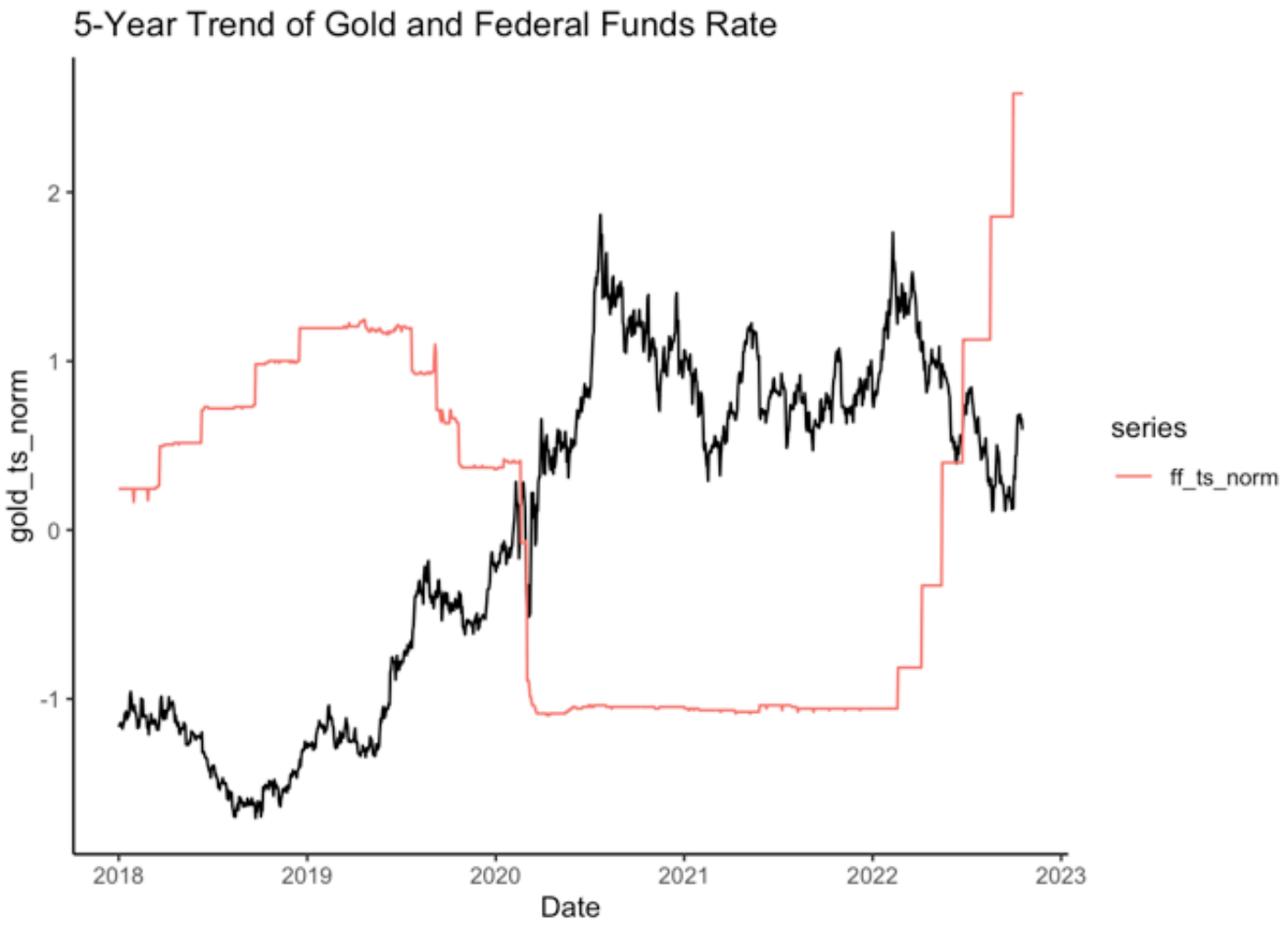
```
#create time series objects for each of the normalized data elements
gold_ts_norm <- ts(full_df_Norm$Gold_Close, start = c(2018, 01), frequency = 252)
unemp_ts_norm <- ts(full_df_Norm$UNRATE, start = c(2018, 01), frequency = 252)
ff_ts_norm <- ts(full_df_Norm$DFF, start = c(2018, 01), frequency = 252)
M2_ts_norm <- ts(full_df_Norm$WM2NS, start = c(2018, 01), frequency = 252)
dj_ts_norm <- ts(full_df_Norm$DJIA, start = c(2018, 01), frequency = 252)
silver_ts_norm <- ts(full_df_Norm$Silver_Close, start = c(2018, 01), frequency = 252)
hits_ts_norm <- ts(full_df_Norm$hits, start = c(2018, 01), frequency = 252)
gold_vol_ts_norm <- ts(full_df_Norm$Gold_Volume, start = c(2018, 01), frequency = 252
)
```

Create time series plots of all factors

```
#line chart of all indicators
autoplot(gold_ts_norm, main = "5-Year Trend of Gold and Unemployment Rate", xlab = "D
ate") +
  autolayer(unemp_ts_norm) + theme_classic()
```

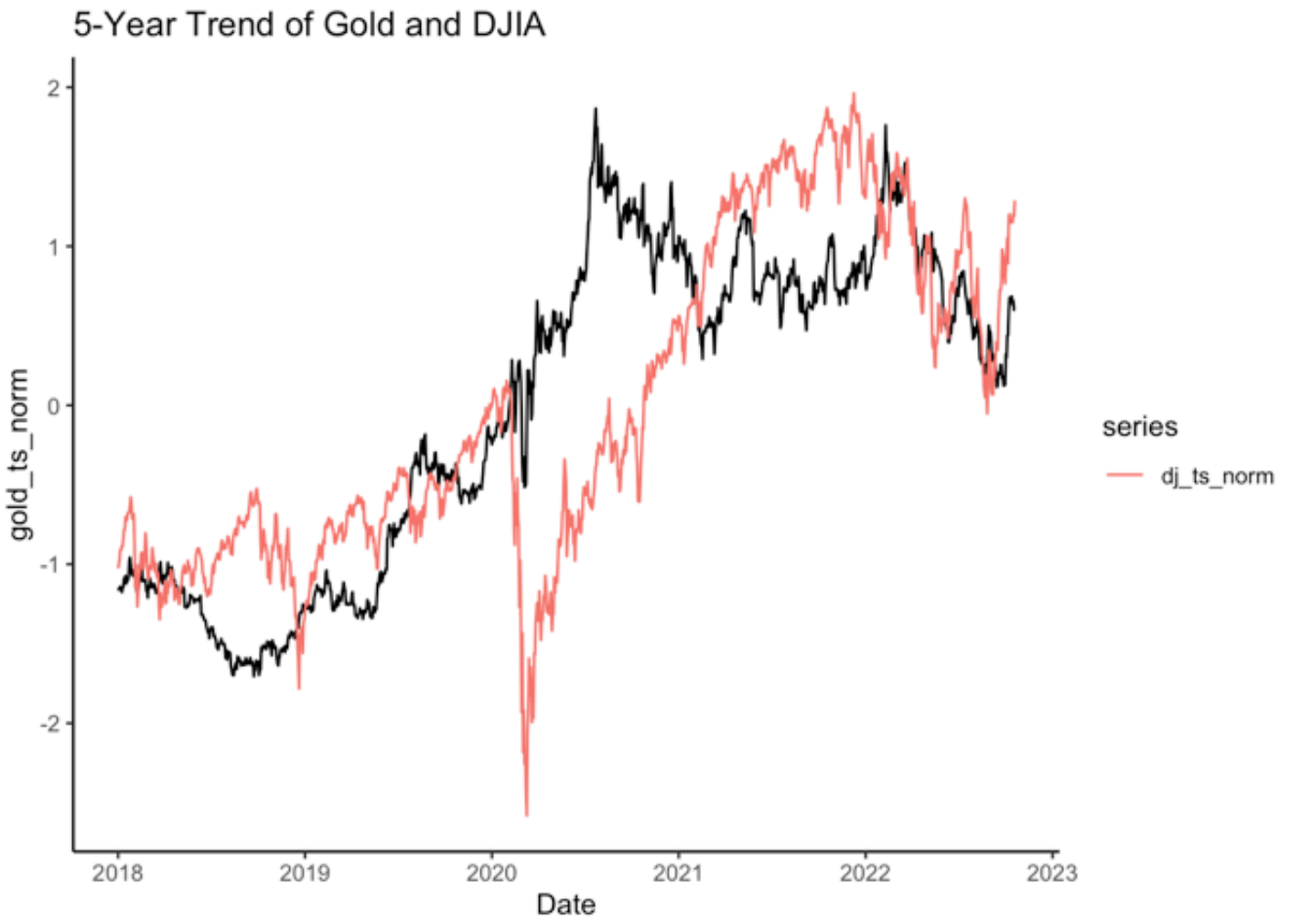



```
autoplot(gold_ts_norm, main = "5-Year Trend of Gold and Federal Funds Rate", xlab = "
Date") +
  autolayer(ff_ts_norm) + theme_classic()
```



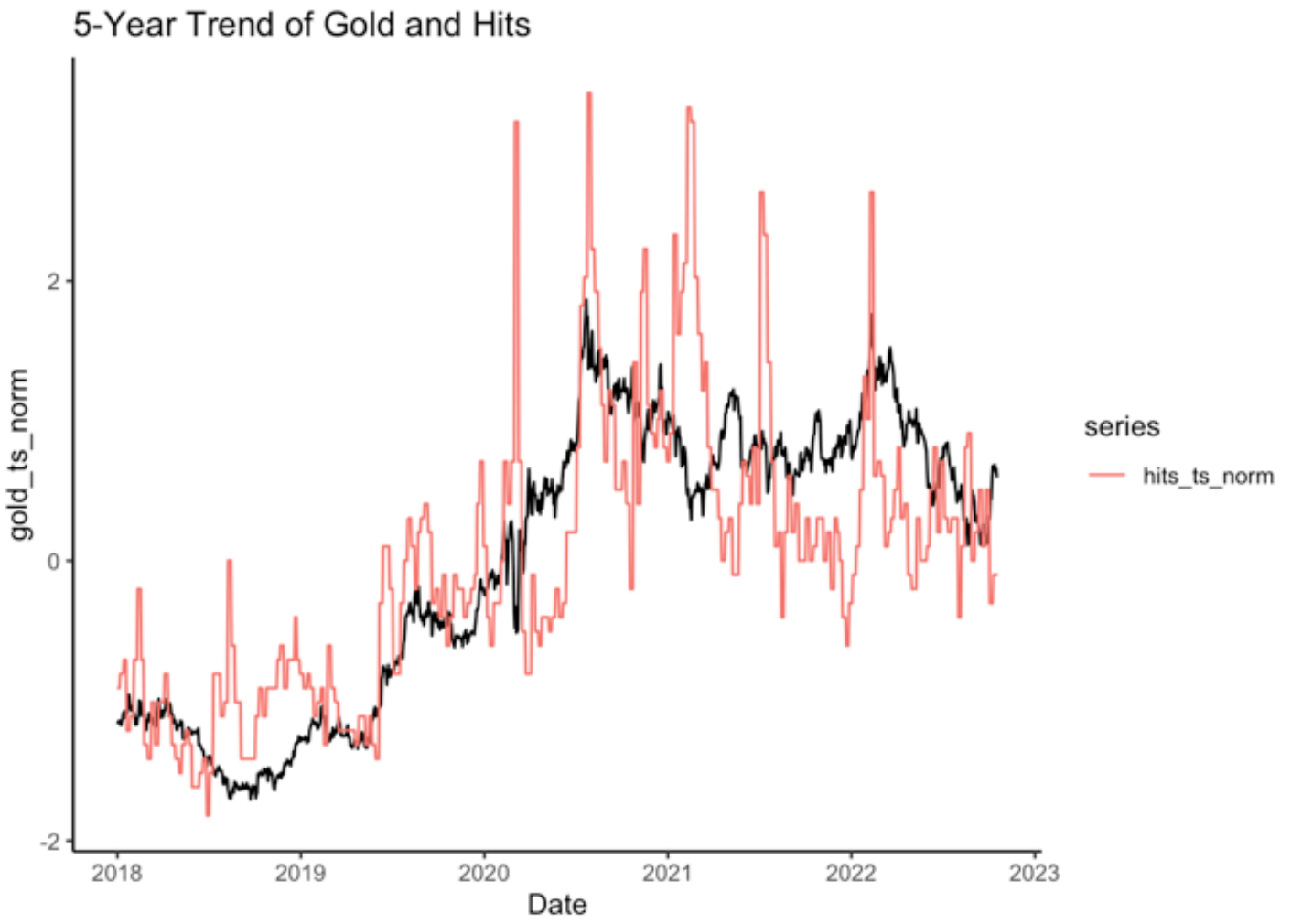

```
autoplot(gold_ts_norm, main = "5-Year Trend of Gold Close and M2", xlab = "Date") +
  autolayer(M2_ts_norm) + theme_classic()
```


```
autoplot(gold_ts_norm, main = "5-Year Trend of Gold and DJIA", xlab = "Date") +
  autolayer(dj_ts_norm) + theme_classic()
```

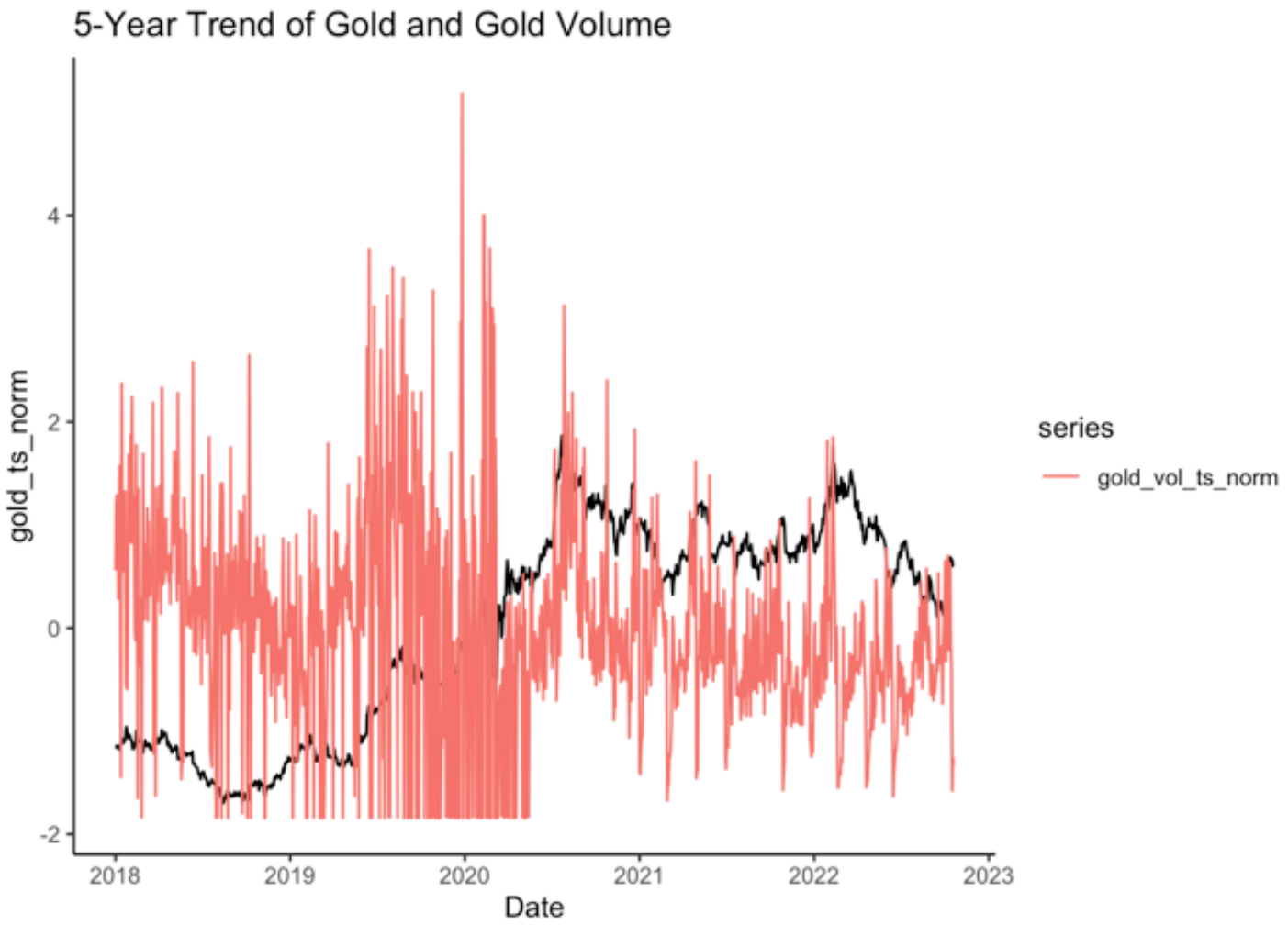



```
autoplot(gold_ts_norm, main = "5-Year Trend of Gold and Silver Close", xlab = "Date")
+
  autolayer(silver_ts_norm) + theme_classic()
```


```
autoplot(gold_ts_norm, main = "5-Year Trend of Gold and Hits", xlab = "Date") +
  autolayer(hits_ts_norm) + theme_classic()
```

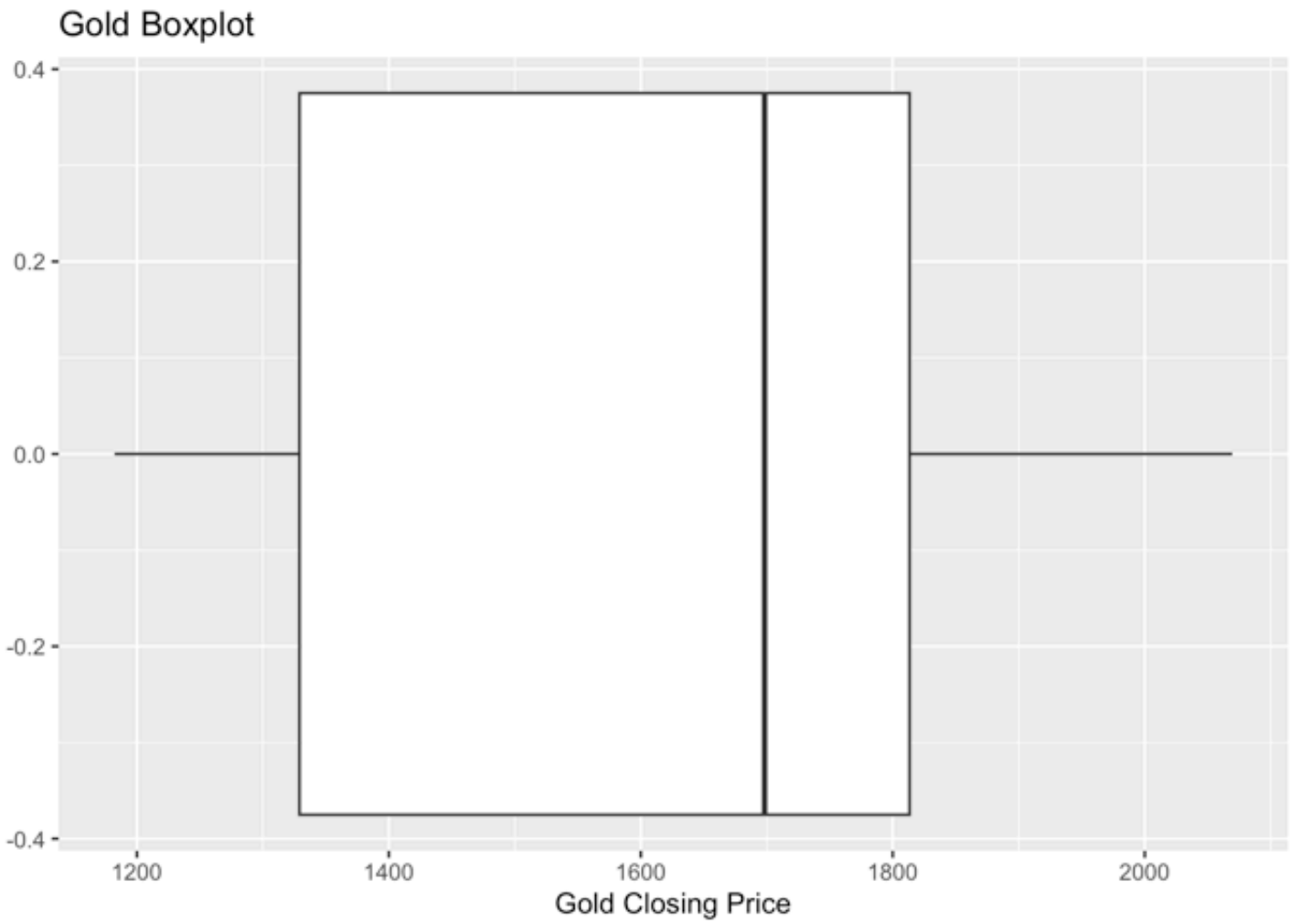



```
autoplot(gold_ts_norm, main = "5-Year Trend of Gold and Gold Volume", xlab = "Date")
+
  autolayer(gold_vol_ts_norm) + theme_classic()
```

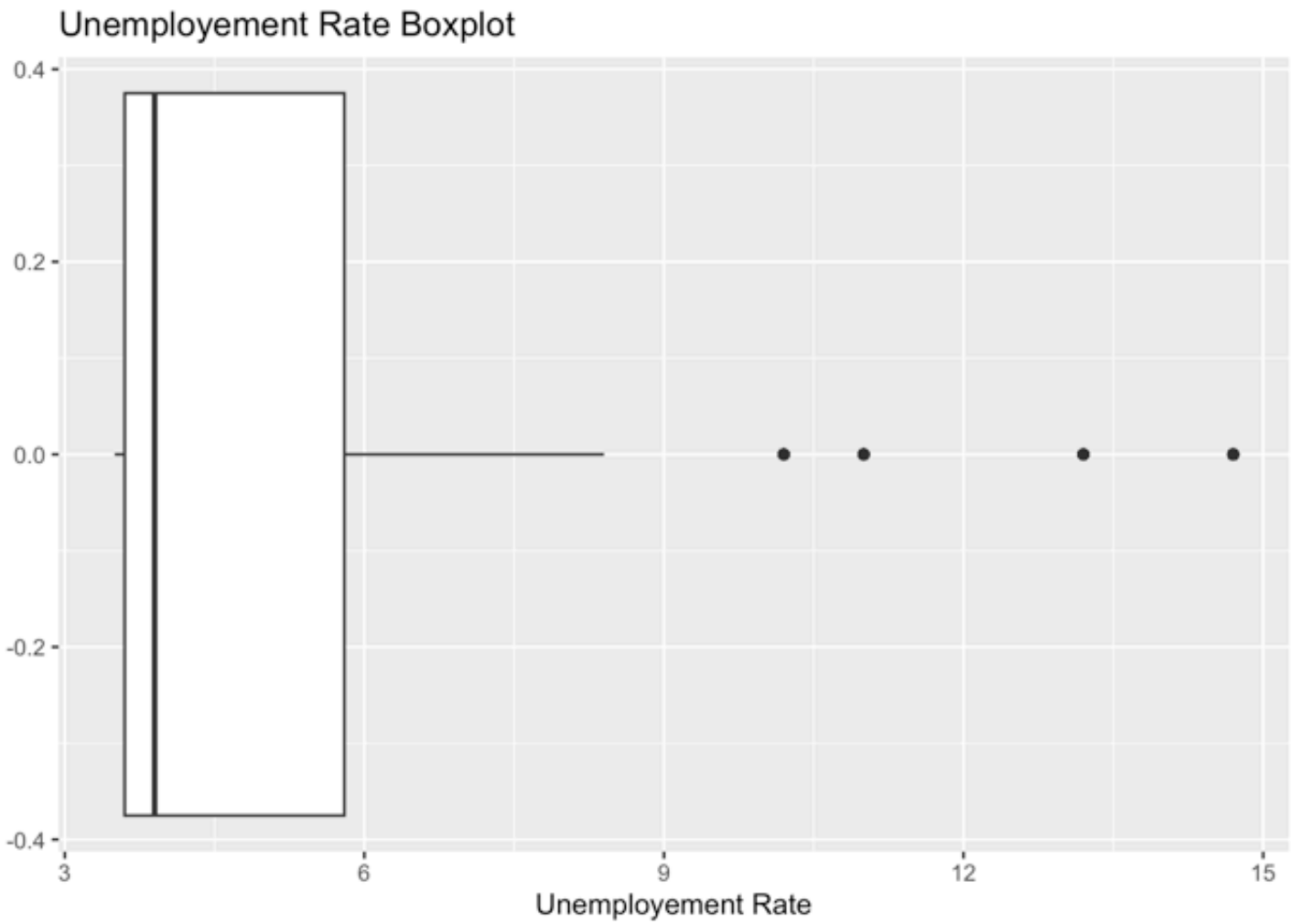


Boxplot to identify outlier variables

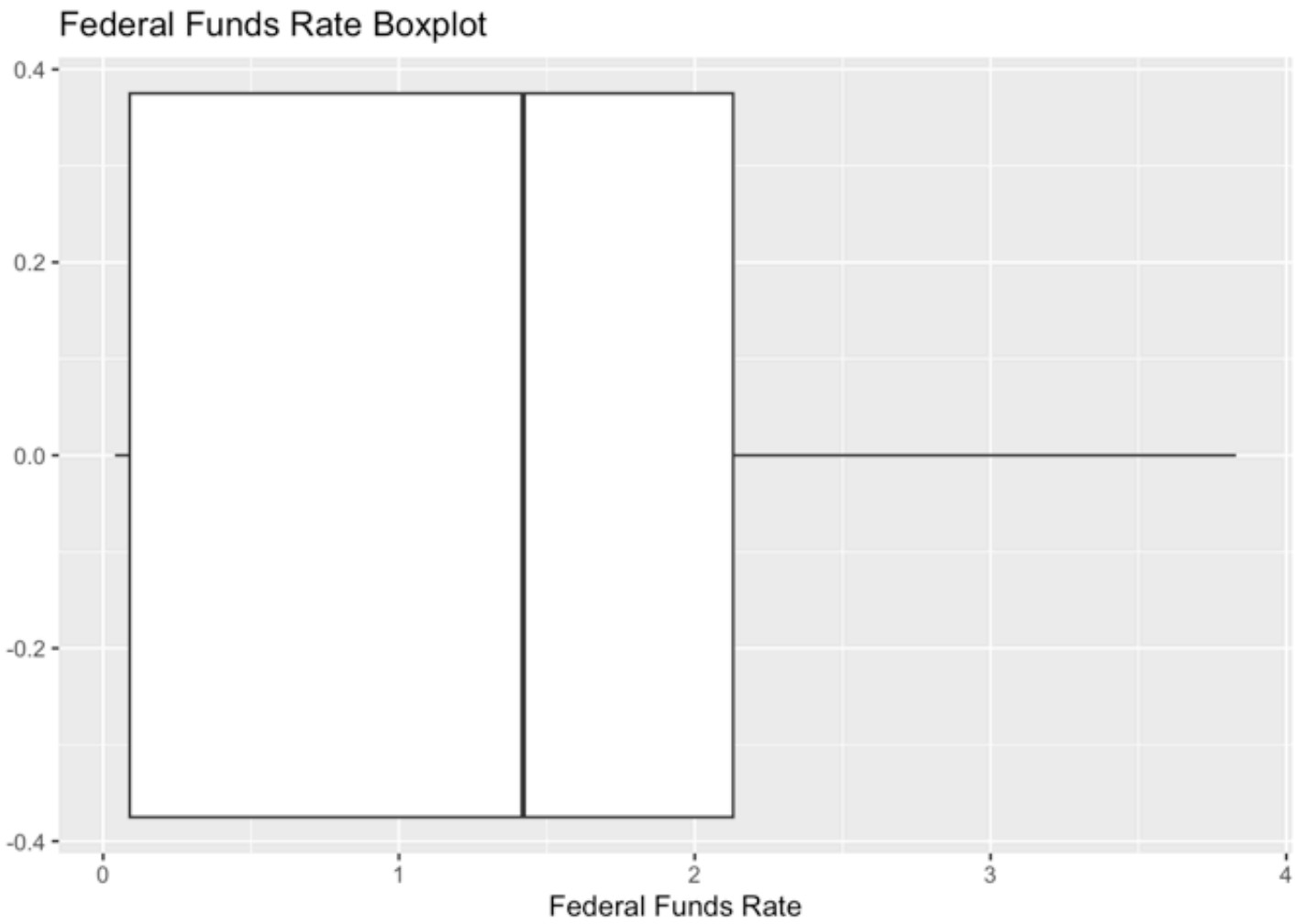
```
#boxplot of all varialbes
ggplot(corr_df, aes(x=Gold_Close)) +
  geom_boxplot() + ggtitle("Gold Boxplot") + xlab("Gold Closing Price")
```



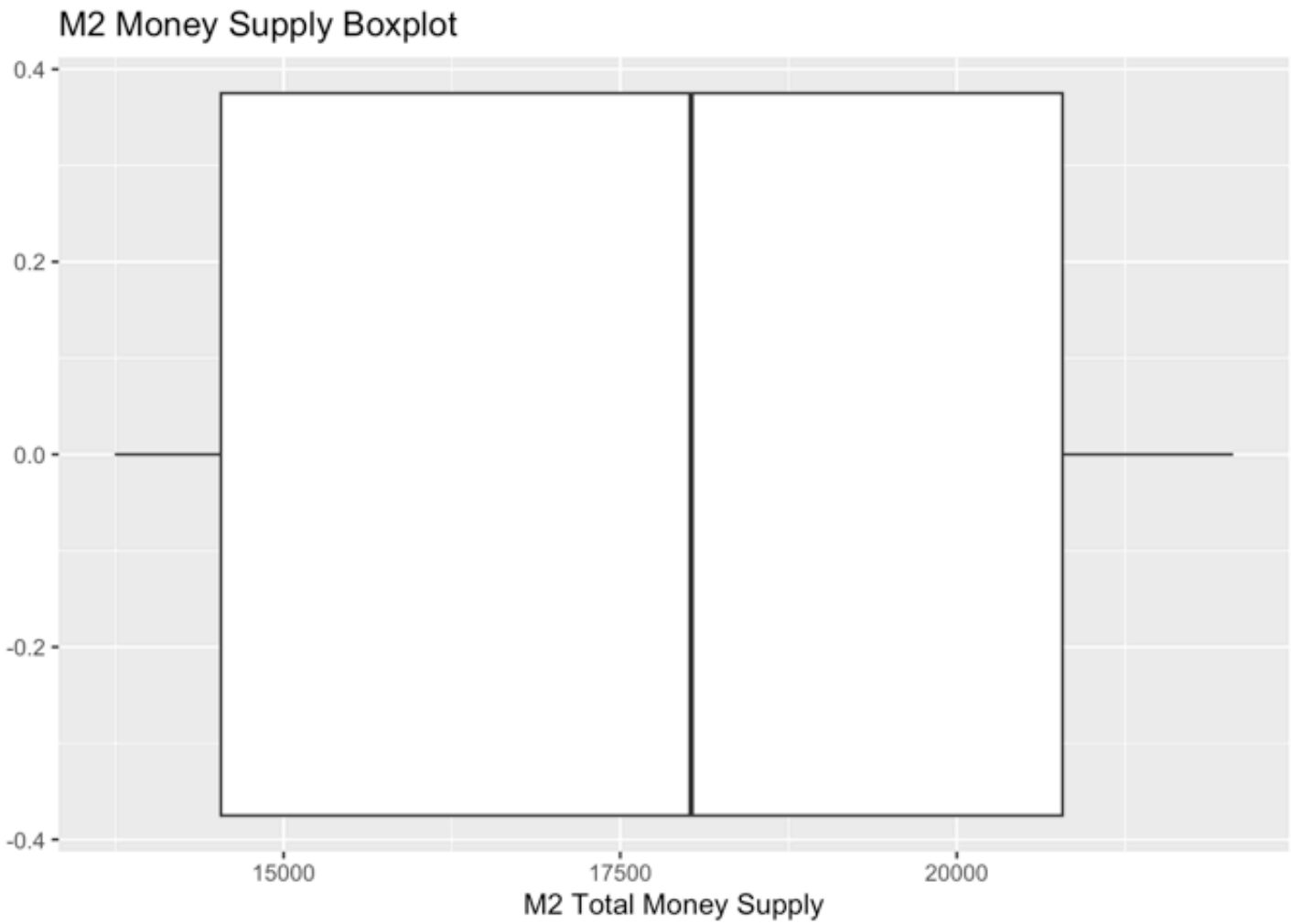

```
ggplot(corr_df, aes(x=UNRATE)) +
  geom_boxplot() + ggtitle("Unemployement Rate Boxplot") + xlab("Unemployement Rate")
```



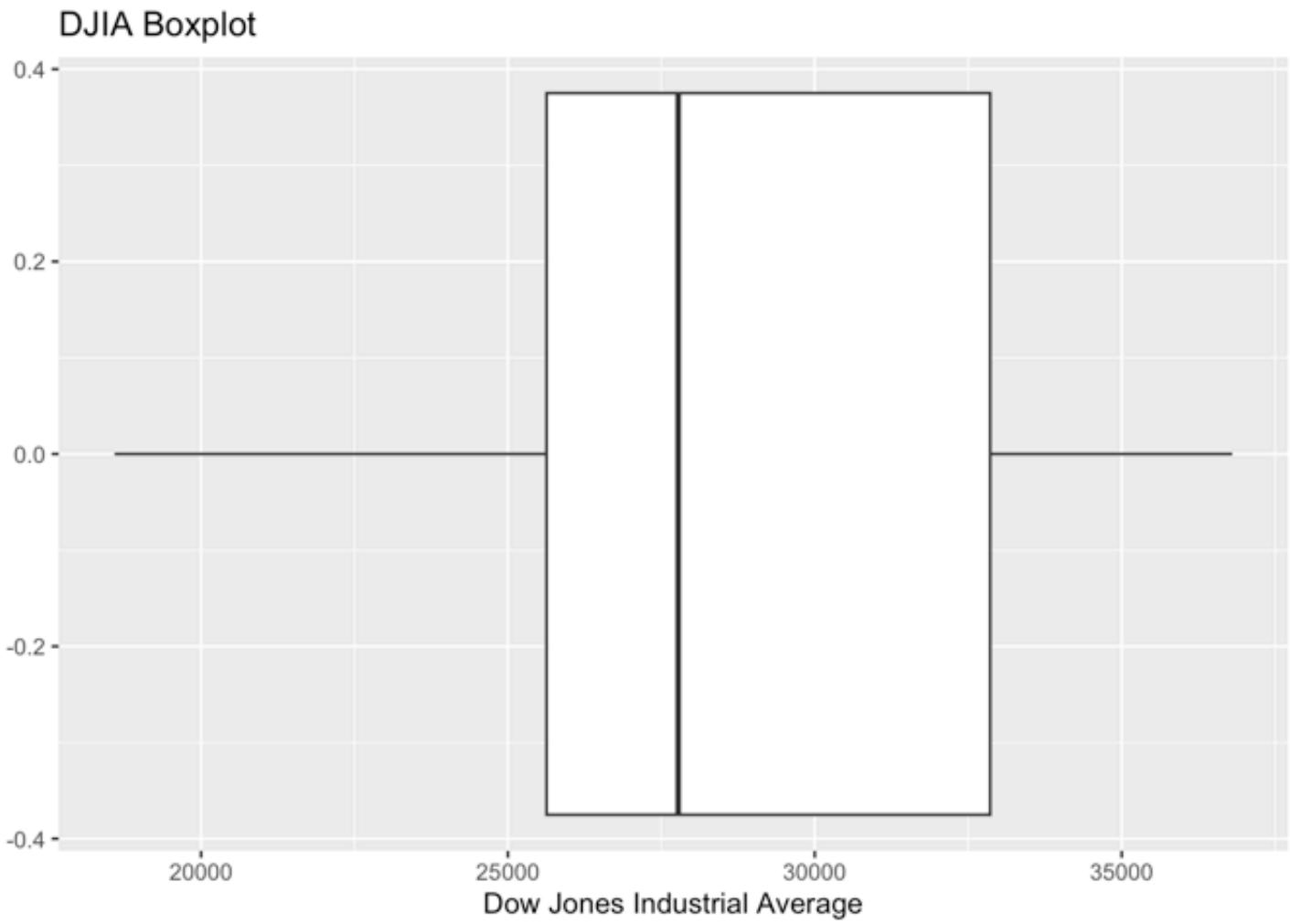

```
ggplot(corr_df, aes(x=DFF)) +
  geom_boxplot() + ggtitle("Federal Funds Rate Boxplot") + xlab("Federal Funds Rate")
```



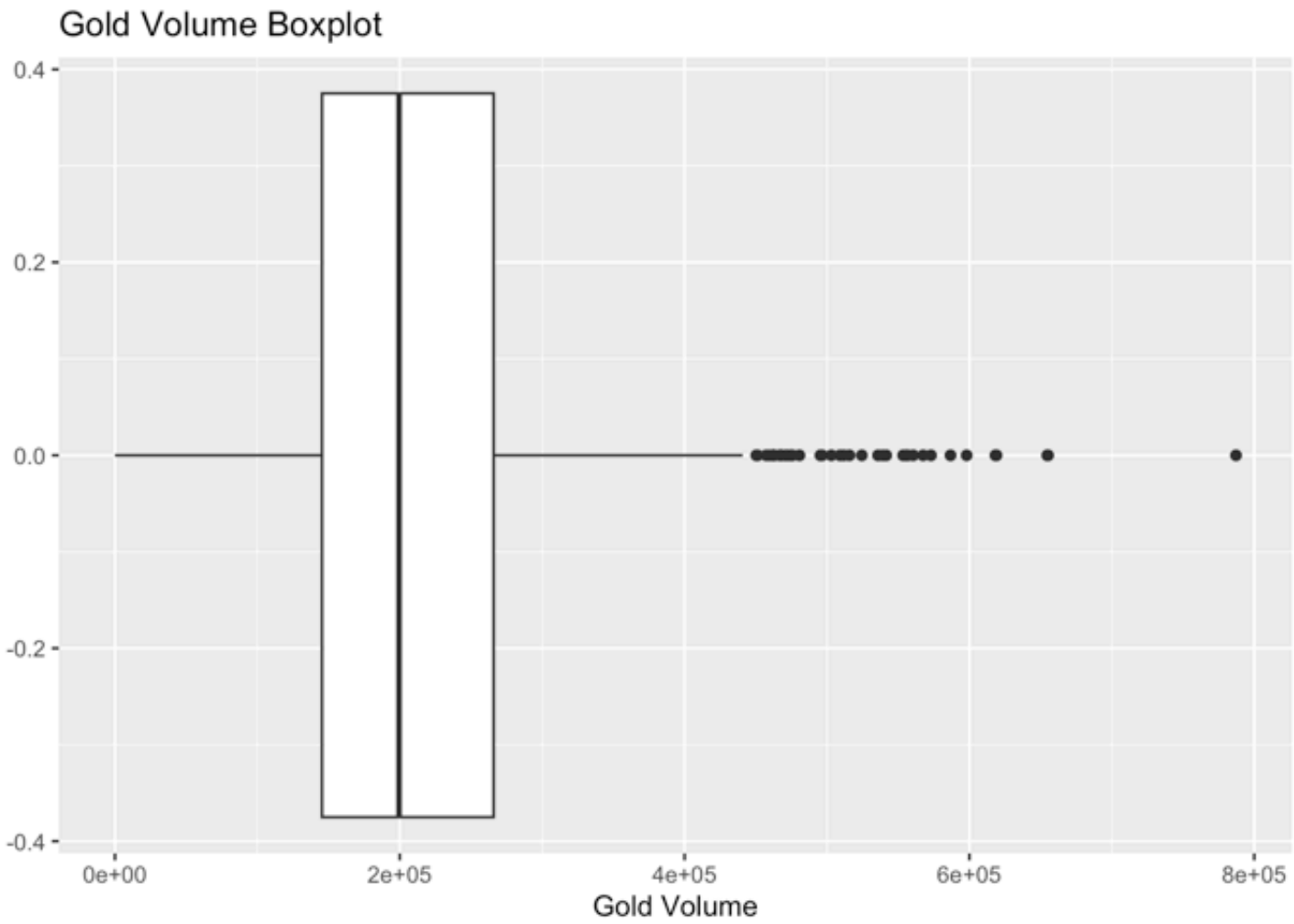

```
ggplot(corr_df, aes(x=WM2NS)) +
  geom_boxplot() + ggtitle("M2 Money Supply Boxplot") + xlab("M2 Total Money Supply")
```



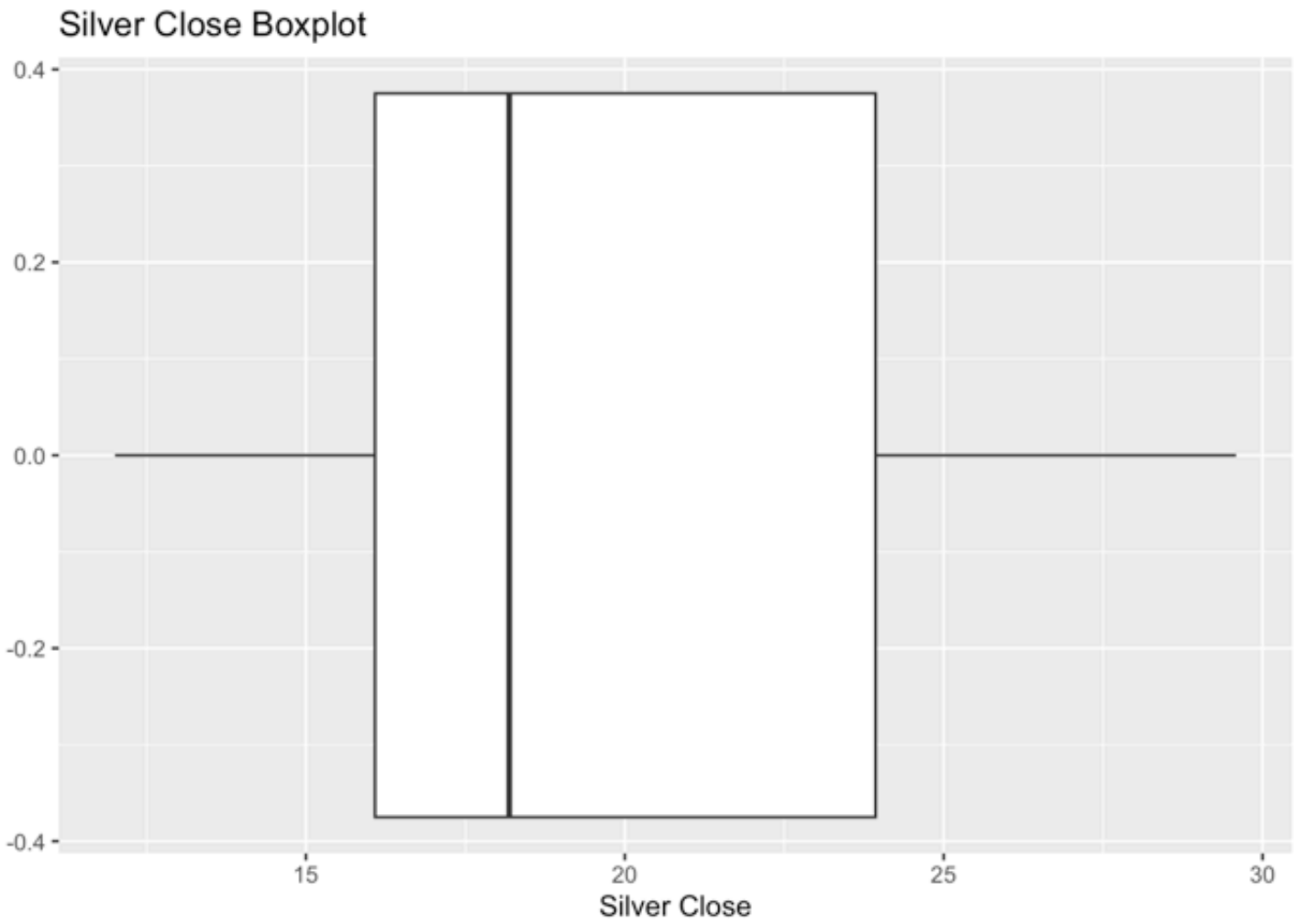

```
ggplot(corr_df, aes(x=DJIA)) +
  geom_boxplot() + ggtitle("DJIA Boxplot") + xlab("Dow Jones Industrial Average")
```



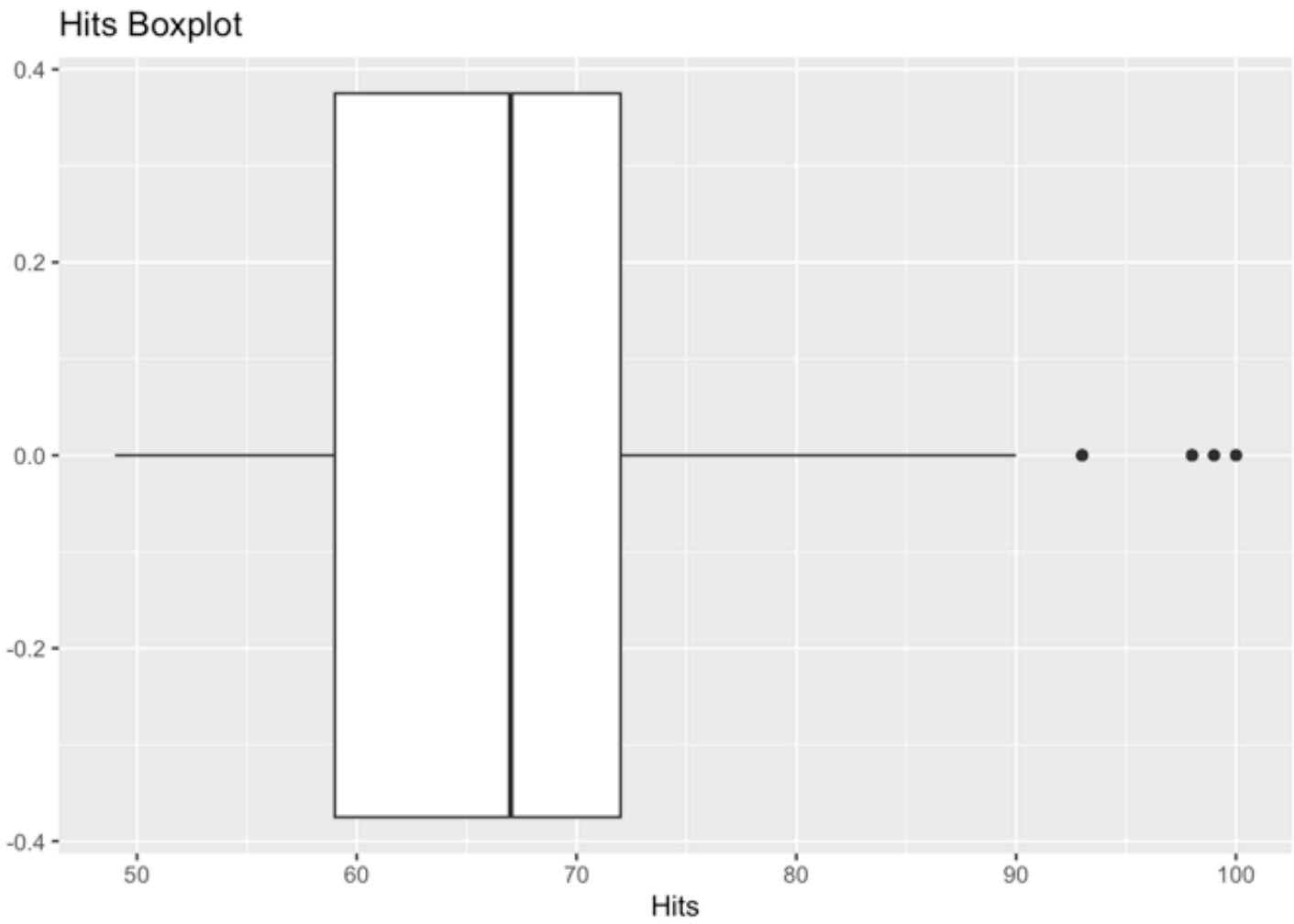

```
ggplot(corr_df, aes(x=Gold_Volume)) +
  geom_boxplot() + ggtitle("Gold Volume Boxplot") + xlab("Gold Volume")
```




```
ggplot(corr_df, aes(x=Silver_Close)) +
  geom_boxplot() + ggtitle("Silver Close Boxplot") + xlab("Silver Close")
```




```
ggplot(corr_df, aes(x=hits)) +
  geom_boxplot() + ggtitle("Hits Boxplot") + xlab("Hits")
```



Full entire Dataset

```
FULL_df  <- left_join(full_df, full_df_Norm, by="Date")
```

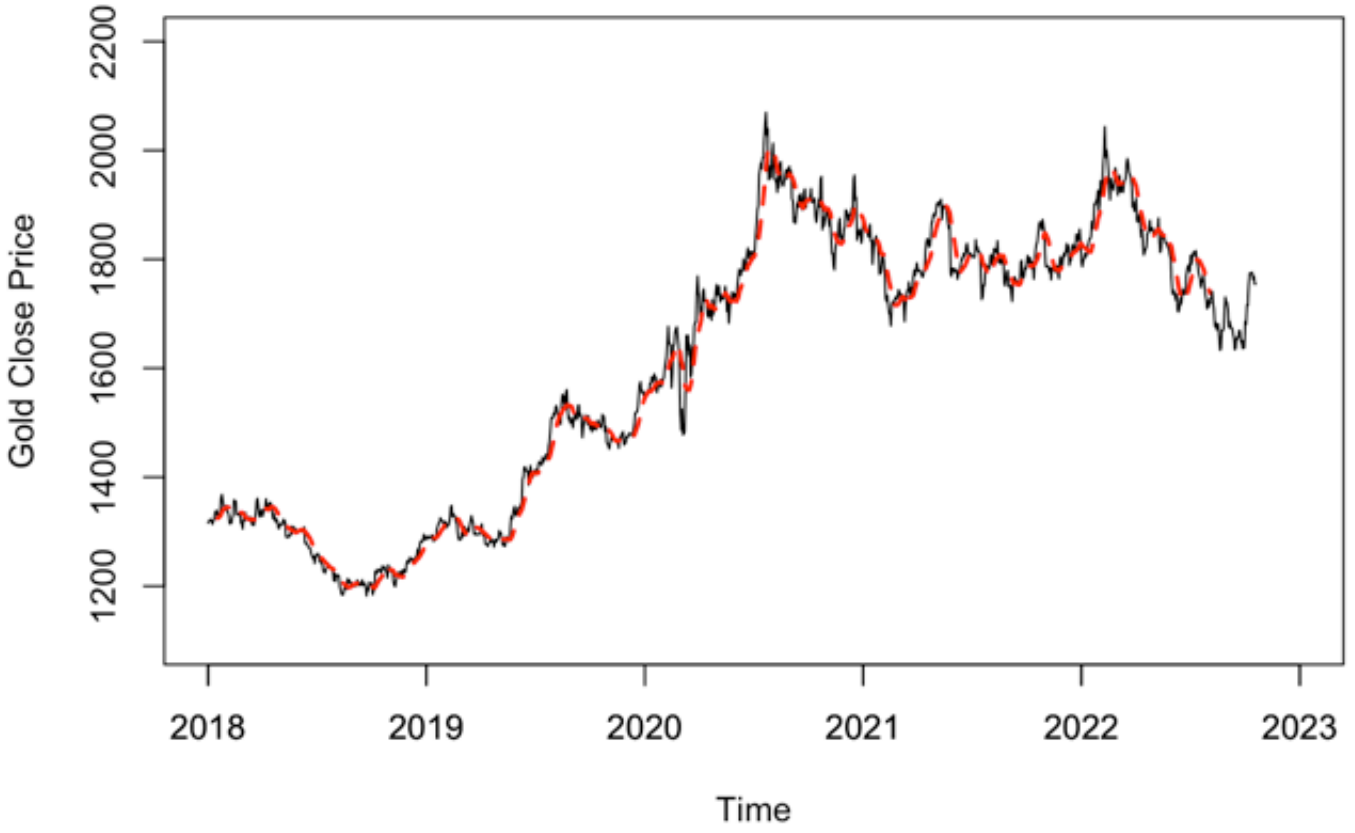
Partition the data into training and testing

```
#create the time series object for the FULL_df
ts_FULL_df <- ts(data=FULL_df$Gold_Close.x, start = c(2018,01), frequency = 252)
train_df <- window(ts_FULL_df, end = c(2022,150)) #161 is end of Aug
test_df <-  window(ts_FULL_df, start = c(2022,151))
#validate the train and test layers connect
autoplot(train_df) + autolayer(test_df) +
  #coord_cartesian(xlim = c(2022,2023)) +
  labs(title = "2022 - 2023 Gold Close Price (Train & Test)",
       x = "Time", y = "Gold Close Price")
```


```
test_df
```

```
## Time Series:
## Start = c(2022, 151)
## End = c(2022, 202)
## Frequency = 252
##  [1] 1728.6 1740.6 1717.4 1709.1 1677.3 1683.5 1671.1 1675.7 1681.1 1655.6
## [11] 1633.4 1636.2 1670.0 1668.6 1672.0 1702.0 1730.5 1720.8 1720.9 1709.3
## [21] 1675.2 1686.0 1677.5 1672.9 1672.9 1664.0 1655.5 1634.2 1636.8 1656.3
## [31] 1654.1 1658.0 1669.2 1668.8 1648.3 1648.3 1636.4 1651.0 1637.7 1685.7
## [41] 1680.5 1716.0 1715.8 1753.7 1774.2 1774.2 1774.7 1775.8 1763.0 1769.0
## [51] 1754.6 1754.8
```

```
#moving average model
nValid <- 12
ma.trailing <- rollmean(train_df, k = 12, align = "right")
last.ma <- tail(ma.trailing, 1)
ma.trailingTEST <- ts(rep(last.ma, nValid), test_df, freq = 12)

plot(train_df, ylim = c(1100, 2200), ylab = "Gold Close Price", xlab = "Time", xlim =
c(2018, 2023), main = "")
axis(1, at = seq(2018, 2023, 1), labels = format(seq(2018, 2023, 1)))
lines(ma.trailing, lwd = 2, col = "red", lty = 2)
lines(ma.trailingTEST, lwd = 2, col = "blue", lty = 2)
lines(test_df)
```




```
summary(ma.trailingTEST)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1737    1737    1737    1737    1737    1737
```

```
#Simple Moving Average with QUANTMOD
getSymbols("GOLD", from = "2018-01-02")
```

```
## [1] "GOLD"
```

```
chartSeries(GOLD)
```


```
addSMA(50)
```


```
ma <- SMA(Cl(GOLD), 50) # ma of adjusted close
accuracy(Cl(GOLD), ma)
```

```
## [1] NA
```

Full entire Dataset

```
FULL_df  <- left_join(full_df, full_df_Norm, by="Date")
FULL_df %>% arrange(mdy(FULL_df$Date))
```

```
## Warning: All formats failed to parse. No formats found.
```

Date	Silver_Close.x	Gold_Close.x	Gold_Volume.x	DX...	UNRAT...	DF...	WM2...
<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2018-01-02	17.060	1316.1	269072.0	91.87	4.0	1.42	13962.7
2018-01-03	17.125	1318.5	342866.0	92.16	4.0	1.42	13962.7
2018-01-04	17.130	1321.6	350803.0	91.85	4.0	1.42	13962.7

2018-01-05	17.155	1322.3	322422.0	91.95	4.0	1.42	13962.7
2018-01-08	17.170	1320.4	238332.0	92.36	4.0	1.42	13969.3
2018-01-09	17.055	1313.7	321636.0	92.53	4.0	1.42	13969.3
2018-01-10	17.135	1319.3	382605.0	92.33	4.0	1.42	13969.3
2018-01-11	17.010	1322.5	254541.0	91.85	4.0	1.42	13969.3
2018-01-12	17.120	1334.9	44651.0	90.97	4.0	1.42	13969.3
2018-01-16	17.095	1337.1	472155.0	90.39	4.0	1.42	13920.3

1-10 of 1,210 rows | 1-10 of 19 columns

Previous **1** 2 3 4 5 6 … 121 Next

```
FULL_df$difference <- c(NA, diff(FULL_df$Gold_Close.x))
FULL_df$diff_boolean <- ifelse(FULL_df$difference >= 0,1,0)
head(FULL_df)
```

	Date <date>	Silver_Close.x <dbl>	Gold_Close.x <dbl>	Gold_Volume.x <dbl>	DX… <dbl>	UNRAT… <dbl>	DF… <dbl>	WM2… <dbl>
1	2018-01-02	17.060	1316.1	269072	91.87	4	1.42	13962.7
2	2018-01-03	17.125	1318.5	342866	92.16	4	1.42	13962.7
3	2018-01-04	17.130	1321.6	350803	91.85	4	1.42	13962.7
4	2018-01-05	17.155	1322.3	322422	91.95	4	1.42	13962.7
5	2018-01-08	17.170	1320.4	238332	92.36	4	1.42	13969.3
6	2018-01-09	17.055	1313.7	321636	92.53	4	1.42	13969.3

6 rows | 1-10 of 22 columns

Partition the data into training and testing

```
#create the time series object for the FULL_df
ts_FULL_df <- ts(data=FULL_df$Gold_Close.x, start = c(2018,01), frequency = 252)

train_df <- window(ts_FULL_df, start = c(2018,01), end = c(2022,186)) #186 is Oct 31
test_df <-  window(ts_FULL_df, start = c(2022,187))

#validate the train and test layers connect
autoplot(train_df) + autolayer(test_df) +
  labs(title = "2022 - 2023 Gold Close Price (Train & Test)",
       x = "Time", y = "Gold Close Price")
```


```
head(train_df)
```

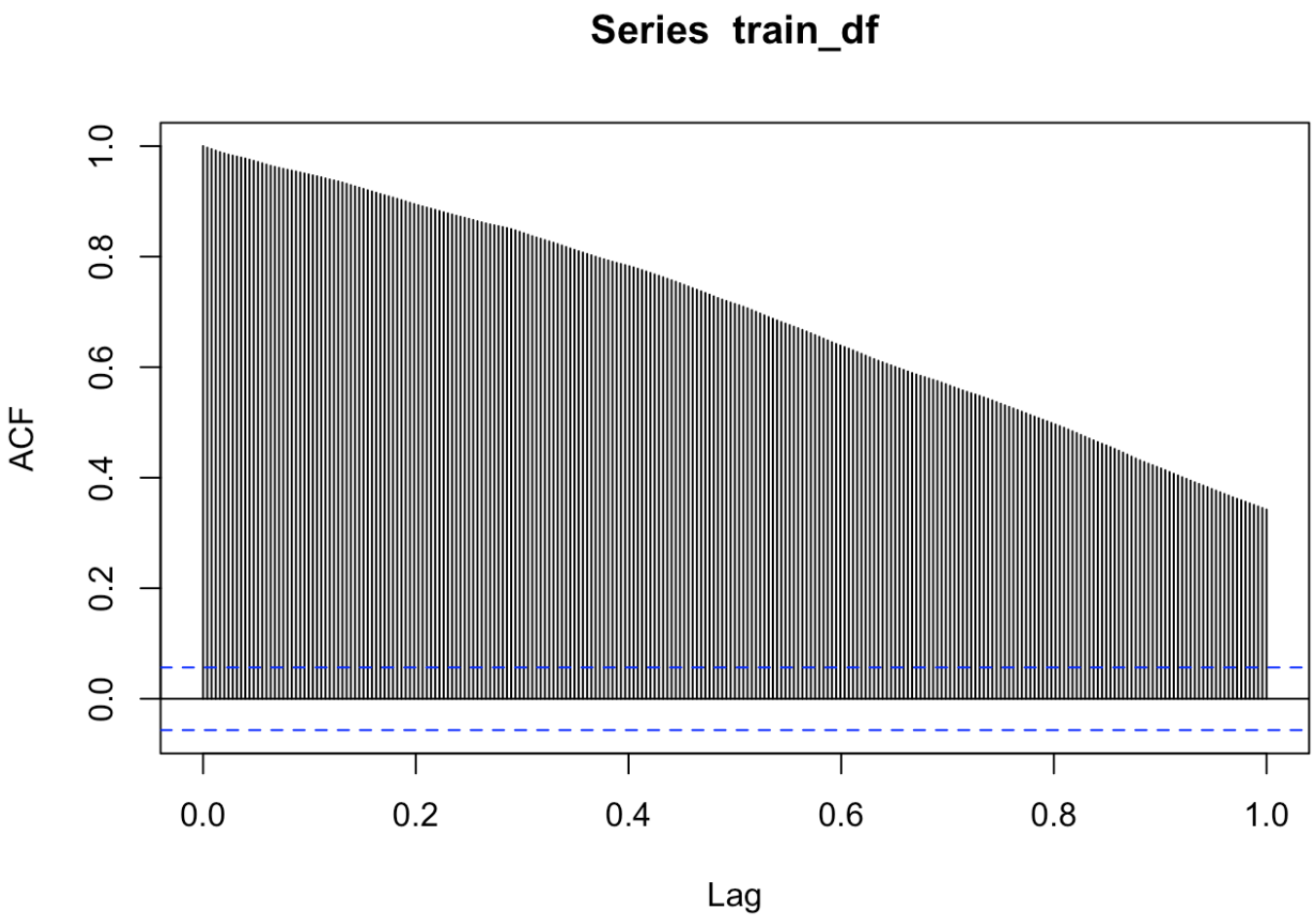
```
## Time Series:
## Start = c(2018, 1)
## End = c(2018, 6)
## Frequency = 252
## [1] 1316.1 1318.5 1321.6 1322.3 1320.4 1313.7
```

Auto ARIMA Model 1 (Gold Close Price Only) - Proof of Random Walk

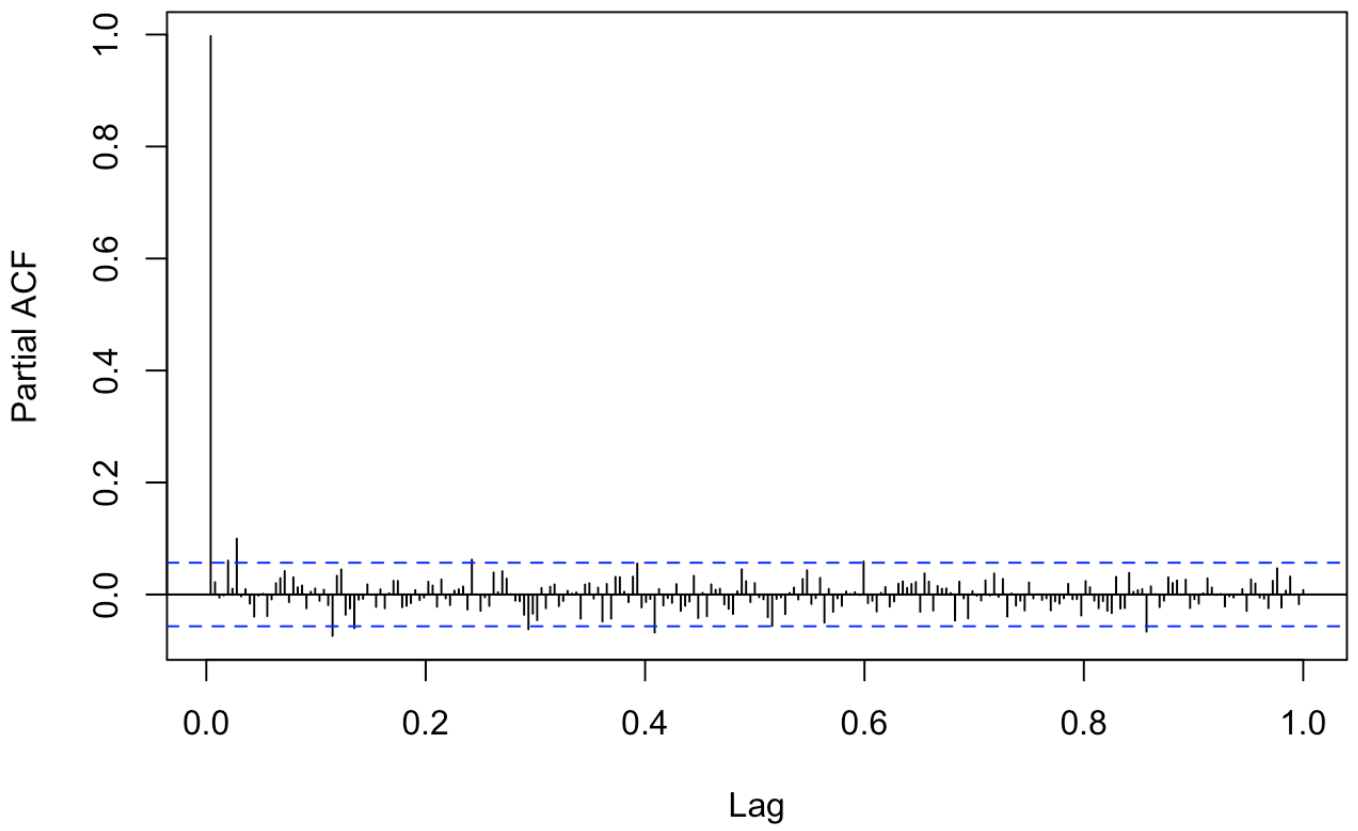
```
#ARIMA Model #1
auto_model <- auto.arima(train_df)
summary(auto_model)
```

```
## Series: train_df
## ARIMA(0,1,0)
##
## sigma^2 = 273.8:  log likelihood = -5040.64
## AIC=10083.28   AICc=10083.28   BIC=10088.36
##
## Training set error measures:
##                      ME    RMSE      MAE        MPE      MAPE       MASE
## Training set 0.2793267 16.5407 11.25923 0.01413727 0.6845585 0.05775767
##                      ACF1
## Training set -0.02760973
```

```
#investigate the potential parameters of the ARIMA model
acf(train_df, lag.max = 252)
```



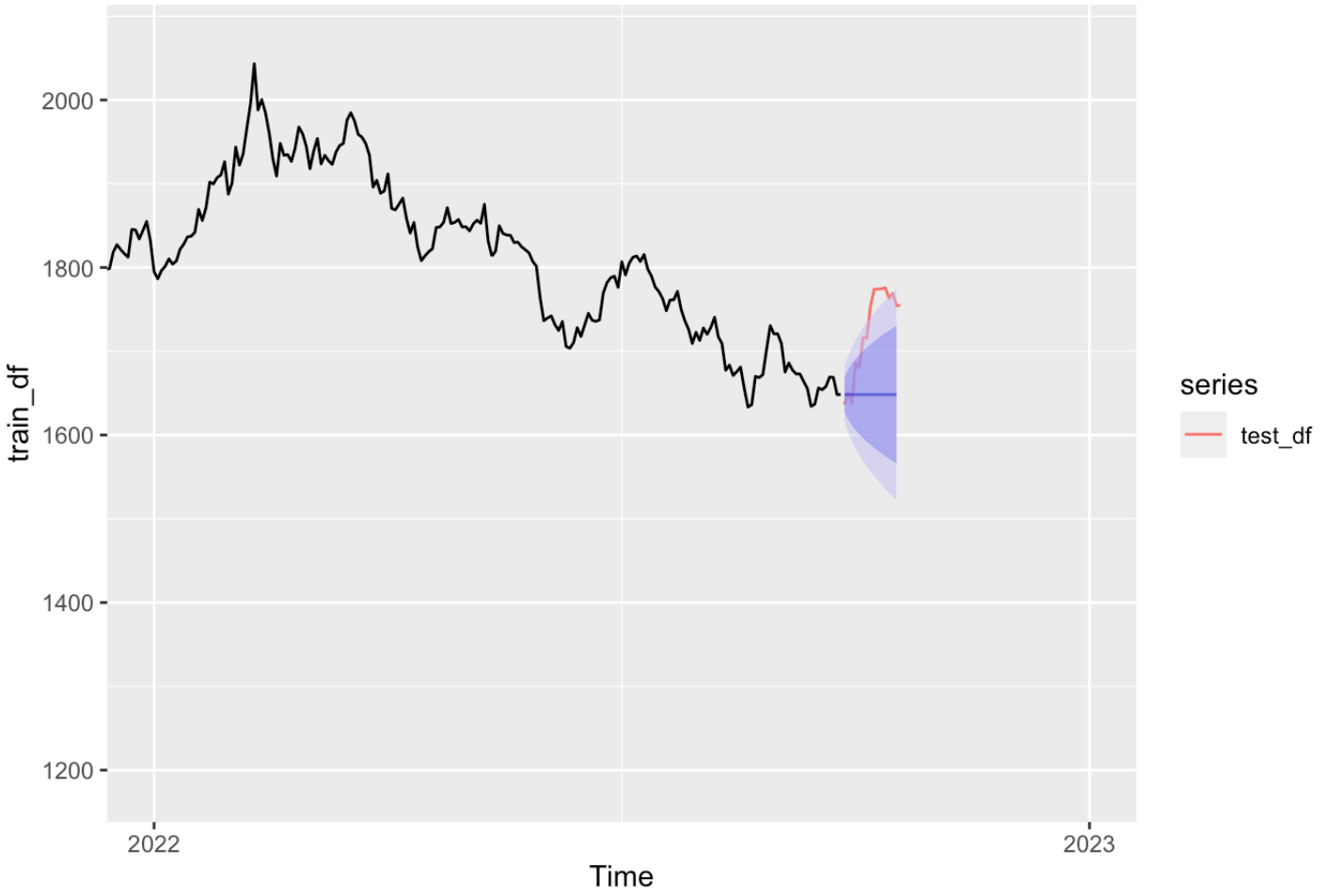

```
pacf(train_df, lag.max = 252)
```




```
#fit to the test data
sqrt(mean(auto_model$residuals^2))
```

```
## [1] 16.5407
```

```
#accuracy(forecast(auto_model, h = 15), test_df)

gold_pred <- forecast(auto_model, h = 15)
#gold_pred

autoplot(train_df) + autolayer(test_df) + autolayer(gold_pred, alpha = 0.5) +coord_ca
rtesian(c(2022,2023))
```



Logistic Regression Model #1 (All Predictors)

```
#logistic regression model # 1

#columns to keep
keep_cols <- c("Date","Gold_Close.x","Silver_Close.x","DXY.x","UNRATE.x","DFF.x", "hi
ts.x","DJIA.x","WM2NS.x", "diff_boolean")
lr_df <- FULL_df[keep_cols]
head(lr_df)
```

	Date <date>	Gold_Close.x <dbl>	Silver_Close.x <dbl>	DX… <dbl>	UNRAT… <dbl>	DF… <dbl>	hits.x <dbl>	DJIA.x <dbl>	WM2… <dbl>
1	2018-01-02	1316.1	17.060	91.87	4	1.42	58	24824.01	13962.7
2	2018-01-03	1318.5	17.125	92.16	4	1.42	58	24922.68	13962.7
3	2018-01-04	1321.6	17.130	91.85	4	1.42	58	25075.13	13962.7
4	2018-01-05	1322.3	17.155	91.95	4	1.42	58	25295.87	13962.7
5	2018-01-08	1320.4	17.170	92.36	4	1.42	59	25283.00	13969.3
6	2018-01-09	1313.7	17.055	92.53	4	1.42	59	25385.80	13969.3

6 rows | 1-10 of 11 columns

```
#create train and test
lr_train <- lr_df[lr_df$Date <= "2022-10-31",]
lr_actual <- lr_df[lr_df$Date > "2022-10-31",]
lr_actual_gold <- lr_df[lr_df$Date > "2022-10-31",]
```

```
#create the logistic regression model with all predictors
set.seed(100)
gold_lr_model <- glm(diff_boolean ~  Silver_Close.x + DXY.x+ UNRATE.x + DFF.x + hits.
x + DJIA.x + WM2NS.x , lr_train,
                     family = "binomial")

#summarize the model output
summary(gold_lr_model)
```

```
##
## Call:
## glm(formula = diff_boolean ~ Silver_Close.x + DXY.x + UNRATE.x +
##     DFF.x + hits.x + DJIA.x + WM2NS.x, family = "binomial", data = lr_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.406  -1.226   1.027   1.122   1.259
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     9.204e-01  2.527e+00   0.364    0.716
## Silver_Close.x -3.914e-02  4.187e-02  -0.935    0.350
## DXY.x          -2.132e-02  3.154e-02  -0.676    0.499
## UNRATE.x        4.921e-02  4.295e-02   1.146    0.252
## DFF.x           4.179e-02  1.644e-01   0.254    0.799
## hits.x          1.145e-02  9.546e-03   1.199    0.230
## DJIA.x          3.725e-05  4.506e-05   0.827    0.408
## WM2NS.x        -4.419e-06  8.841e-05  -0.050    0.960
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1647.2  on 1192  degrees of freedom
## Residual deviance: 1642.5  on 1185  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 1658.5
##
## Number of Fisher Scoring iterations: 4
```

```
gold_lr_model
```

```
##
## Call:  glm(formula = diff_boolean ~ Silver_Close.x + DXY.x + UNRATE.x +
##     DFF.x + hits.x + DJIA.x + WM2NS.x, family = "binomial", data = lr_train)
##
## Coefficients:
##    (Intercept)  Silver_Close.x           DXY.x        UNRATE.x           DFF.x
##      9.204e-01      -3.914e-02      -2.132e-02       4.921e-02       4.179e-02
##         hits.x          DJIA.x         WM2NS.x
##      1.145e-02       3.725e-05      -4.419e-06
##
## Degrees of Freedom: 1192 Total (i.e. Null);  1185 Residual
##   (1 observation deleted due to missingness)
## Null Deviance:       1647
## Residual Deviance: 1643  AIC: 1659
```

```
#generate the predictions of the lr model
gold_pred_prob <- predict(gold_lr_model, newdata = lr_actual, type = "response")
gold_pred_df <- cbind(lr_actual, gold_pred_prob)

#identify columns to keep and create the gold boolean flag
eval_cols <- c("Date","diff_boolean","gold_pred_prob")
eval_df <- gold_pred_df[eval_cols]
eval_df$gold_boolean <- ifelse(eval_df$gold_pred_prob >= .50,1,0)

eval_df <- cbind(eval_df, lr_actual_gold$Gold_Close.x)
eval_df
```

	Date	diff_boolean	gold_pred_prob	gold_boolean	lr_actual_gold$Gold_Clo
	<date>	<dbl>	<dbl>	<dbl>	<
1195	2022-11-01	0	0.5020122	1	16
1196	2022-11-02	1	0.5001512	1	16
1197	2022-11-03	0	0.5066160	1	16
1198	2022-11-04	1	0.5110586	1	16
1199	2022-11-07	0	0.4892661	0	16
1200	2022-11-08	1	0.4941486	0	17
1201	2022-11-09	0	0.4777125	0	17
1202	2022-11-10	1	0.5036066	1	17
1203	2022-11-11	1	0.5103247	1	17
1204	2022-11-14	1	0.5122070	1	17

```
#create the confusion matrix for the LR model with all predictors
confusionMatrix(data = as.factor(eval_df$diff_boolean), reference = as.factor(eval_df
$gold_boolean), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 2 4
##          1 1 9
##
##                Accuracy : 0.6875
##                  95% CI : (0.4134, 0.8898)
##     No Information Rate : 0.8125
##     P-Value [Acc > NIR] : 0.9373
##
##                   Kappa : 0.2593
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.6923
##             Specificity : 0.6667
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.3333
##              Prevalence : 0.8125
##          Detection Rate : 0.5625
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.6795
##
##        'Positive' Class : 1
##
```

Logistic Regresson Model #2 (with highly correlated features)

```
#create the data frame of values with high correlation
high_corr_feat <- c("Date","Silver_Close.x","WM2NS.x","hits.x","DFF.x","diff_boolean"
)
lr_df2 <- FULL_df[high_corr_feat]
#head(lr_df2)
```

```
#create train and test sets
lr_train2 <- lr_df2[lr_df2$Date <= "2022-10-31",]
lr_actual2 <- lr_df2[lr_df2$Date > "2022-10-31",]
```

```
#create the logistic regression model with high only correlated predictors
set.seed(100)
#gold_lr_model2 <- glm(diff_boolean ~  Silver_Close.x  + DFF.x + hits.x + DJIA.x + WM
2NS.x , lr_train2,
#                     family = "binomial")
gold_lr_model2 <- glm(diff_boolean ~  Silver_Close.x   + WM2NS.x , lr_train2,
                    family = "binomial")
#summarize the model output
summary(gold_lr_model2)
```

```
##
## Call:
## glm(formula = diff_boolean ~ Silver_Close.x + WM2NS.x, family = "binomial",
##     data = lr_train2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.292  -1.236   1.081   1.119   1.154
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.055e-02  3.433e-01   0.118    0.906
## Silver_Close.x  1.420e-02  2.150e-02   0.661    0.509
## WM2NS.x        -9.784e-06  3.096e-05  -0.316    0.752
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1647.2  on 1192  degrees of freedom
## Residual deviance: 1646.7  on 1190  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: 1652.7
##
## Number of Fisher Scoring iterations: 3
```

```
gold_lr_model2
```

```
##
## Call:  glm(formula = diff_boolean ~ Silver_Close.x + WM2NS.x, family = "binomial",
##     data = lr_train2)
##
## Coefficients:
##    (Intercept)  Silver_Close.x         WM2NS.x
##      4.055e-02       1.420e-02      -9.784e-06
##
## Degrees of Freedom: 1192 Total (i.e. Null);  1190 Residual
##   (1 observation deleted due to missingness)
## Null Deviance:       1647
## Residual Deviance: 1647  AIC: 1653
```

```
#generate the predictions of the lr_model2
gold_pred_prob2 <- predict(gold_lr_model2, newdata = lr_actual2, type = "response")
gold_pred_df2 <- cbind(lr_actual2, gold_pred_prob2)

#identify columns to keep and create the gold boolean flag
eval_cols2 <- c("Date","diff_boolean","gold_pred_prob2")
eval_df2 <- gold_pred_df2[eval_cols2]
eval_df2$gold_boolean <- ifelse(eval_df2$gold_pred_prob >= .50,1,0)

eval_df2
```

	Date	diff_boolean	gold_pred_prob2	gold_boolean
	<date>	<dbl>	<dbl>	<dbl>
1195	2022-11-01	0	0.5289099	1
1196	2022-11-02	1	0.5281313	1
1197	2022-11-03	0	0.5250861	1
1198	2022-11-04	1	0.5287861	1
1199	2022-11-07	0	0.5312804	1
1200	2022-11-08	1	0.5315634	1
1201	2022-11-09	0	0.5335788	1
1202	2022-11-10	1	0.5327657	1
1203	2022-11-11	1	0.5341444	1
1204	2022-11-14	1	0.5341091	1

1-10 of 16 rows Previous **1** 2 Next

The Logistic Regression model with only highly correlated features predicts all positive cases for the gold_boolean flag

```
#return the distinct values of gold Boolean
n_distinct(eval_df2$gold_boolean)
```

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

Function to find local min and max for buy/sell signals

```
 # Locate Local Min and Max for buy/sell signals
locate_xtrem <- function (x, last = FALSE)
{
  # use rle to deal with duplicates
  x_rle <- rle(x)

  # force the first value to be identified as an extrema
  first_value <- x_rle$values[1] - x_rle$values[2]

  #
  # ! NOTE: with this method, last value will be considered as an extrema
  diff_sign_rle <- c(first_value, diff(x_rle$values)) %>% sign() %>% rle()

  # this vector will be used to get the initial positions
  diff_idx <- cumsum(diff_sign_rle$lengths)

  # find min and max
  diff_min <- diff_idx[diff_sign_rle$values < 0]
  diff_max <- diff_idx[diff_sign_rle$values > 0]

  # get the min and max indexes in the original series
  x_idx <- cumsum(x_rle$lengths)
  if (last) {
    min <- x_idx[diff_min]
    max <- x_idx[diff_max]
  } else {
    min <- x_idx[diff_min] - x_rle$lengths[diff_min] + 1
    max <- x_idx[diff_max] - x_rle$lengths[diff_max] + 1
  }
  # just get number of occurences
  min_nb <- x_rle$lengths[diff_min]
  max_nb <- x_rle$lengths[diff_max]

  # format the result as a tibble
  bind_rows(
    tibble(Idx = min, Values = x[min], NB = min_nb, Status = "min"),
    tibble(Idx = max, Values = x[max], NB = max_nb, Status = "max")) %>%
    arrange(.data$Idx) %>%
    mutate(Last = last) %>%
    mutate_at(vars(.data$Idx, .data$NB), as.integer)
}
```

GOLD Prophet FORECAST BASELINE (No External Regressors)

```
# Partition Data frame
prophet.Train <- subset(FULL_df, Date < as.Date("2022-11-08"))
Prophet.Test <- subset(FULL_df, Date >= as.Date("2022-11-17"))

# Set training col for prophet forecast
prophet.Train <- prophet.Train %>%
  select(c("Date","Gold_Close.x")) %>%
  rename(ds = Date, y = Gold_Close.x)

# Prophet Predictions
Prophet <- prophet(prophet.Train, interval.width = 0.95)
```

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

```
future <- make_future_dataframe(Prophet, periods = 7) %>% filter(!wday(ds) %in% c(1,7
)) #account for regular gaps on weekends
Prophet_Forecast_base <- predict(Prophet, future)

# Grab necessary variables
Forecast_subset <- Prophet_Forecast_base %>%
  select(c('ds','yhat','yhat_lower','yhat_upper')) %>%
  rename(Date = ds, ClosePrice = yhat , ClosePrice_lower = yhat_lower, ClosePrice_upp
er = yhat_upper)

# Put into Dataframe
datatable(Forecast_subset[c('Date','ClosePrice','ClosePrice_lower','ClosePrice_upper'
)])
```

Show 10 entries

Search:

	Date	ClosePrice	ClosePrice_lower	ClosePrice_upper
1	2018-01-02T00:00:00Z	1311.63750243046	1247.93996694001	1381.01760672916
2	2018-01-03T00:00:00Z	1312.78369479746	1242.5987268899	1374.00124377786
3	2018-01-04T00:00:00Z	1314.3140227536	1251.07406546917	1383.05403809662
4	2018-01-05T00:00:00Z	1313.50038859346	1249.87168131471	1376.48932533478
5	2018-01-08T00:00:00Z	1315.26168274397	1250.27349150937	1383.04978865276
6	2018-01-09T00:00:00Z	1316.30627861412	1250.90511521725	1379.56533014966
7	2018-01-10T00:00:00Z	1316.63524291449	1249.75806123659	1384.19420117904

8	2018-01-11T00:00:00Z	1317.48061754534	1253.48420655579	1383.56520026427
9	2018-01-12T00:00:00Z	1316.11373602952	1251.79751614467	1378.72919901756
10	2018-01-16T00:00:00Z	1317.89995881398	1253.50883370456	1386.05506926296

Showing 1 to 10 of 1,204 entries Previous 1 2 3 4 5 … 121 Next

```
# Return prediction results from prophet forecast
Prophet_Results_base <- Prophet_Forecast_base %>%
  select(c("ds","yhat")) %>%
  rename(Date = ds, Close_Prediction = yhat)
Prophet_Results_base$Date <- as.Date(Prophet_Results_base$Date , format = "%m/%d/%y")
Prophet_Results_base <- subset(Prophet_Results_base, Date >= as.Date("2022-11-08"))

# Grab actual gold data
Gold_Results <- gold %>%
  select(c("Date","Gold_Close"))  %>%
  rename(Close_Actual = Gold_Close)

# Join prediction and actual results for comparison
results <- left_join(Prophet_Results_base, Gold_Results,  by="Date")
results
```

Date <date>	Close_Prediction <dbl>	Close_Actual <dbl>
2022-11-08	1676.151	1716.0
2022-11-09	1676.111	1715.8
2022-11-10	1676.556	1753.7
2022-11-11	1674.727	1774.2
2022-11-14	1673.528	1774.2

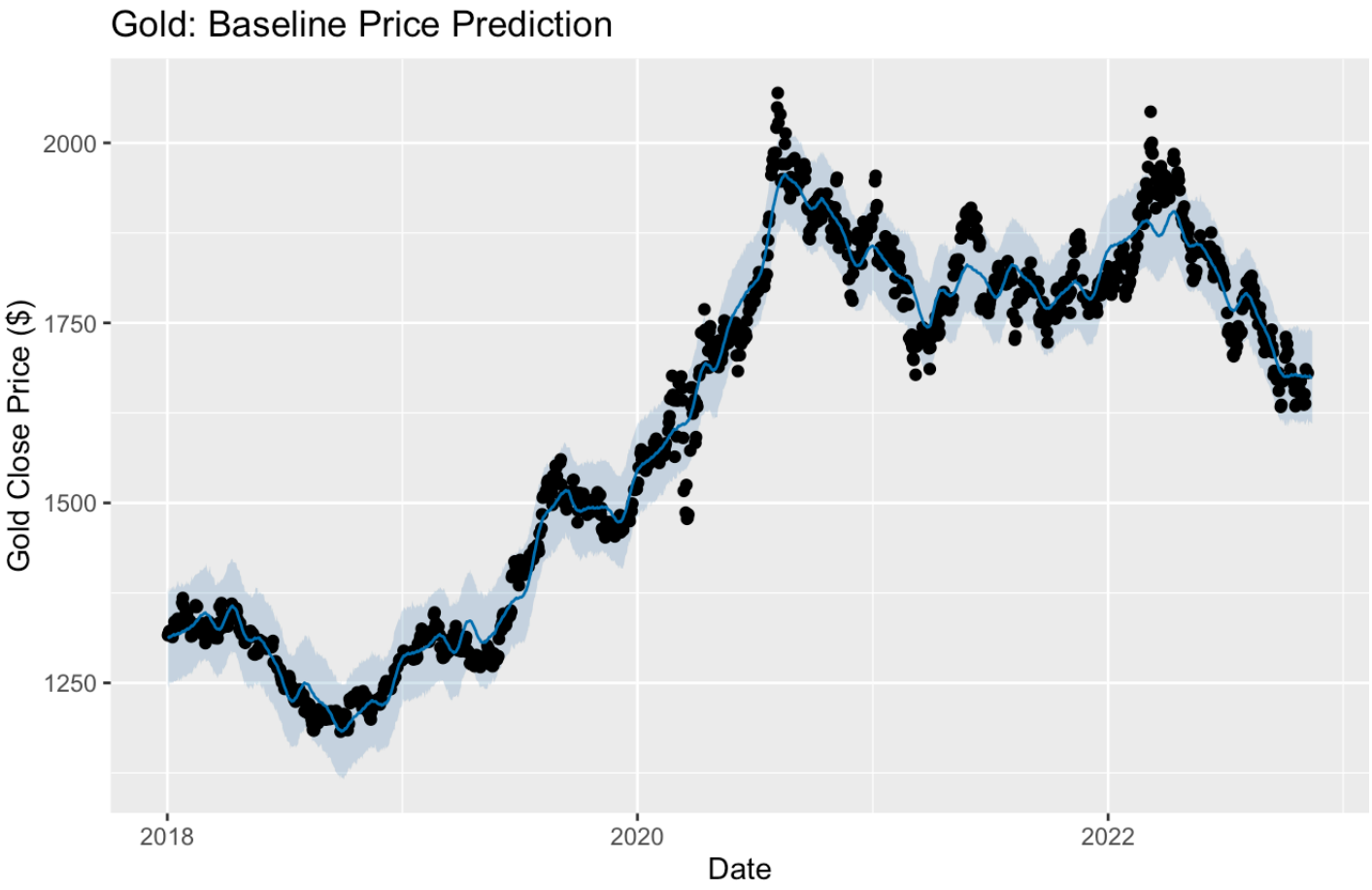
5 rows

```
# Calculate RMSE
RMSE <- sqrt(mean((results$Close_Actual - results$Close_Prediction)^2))
RMSE
```

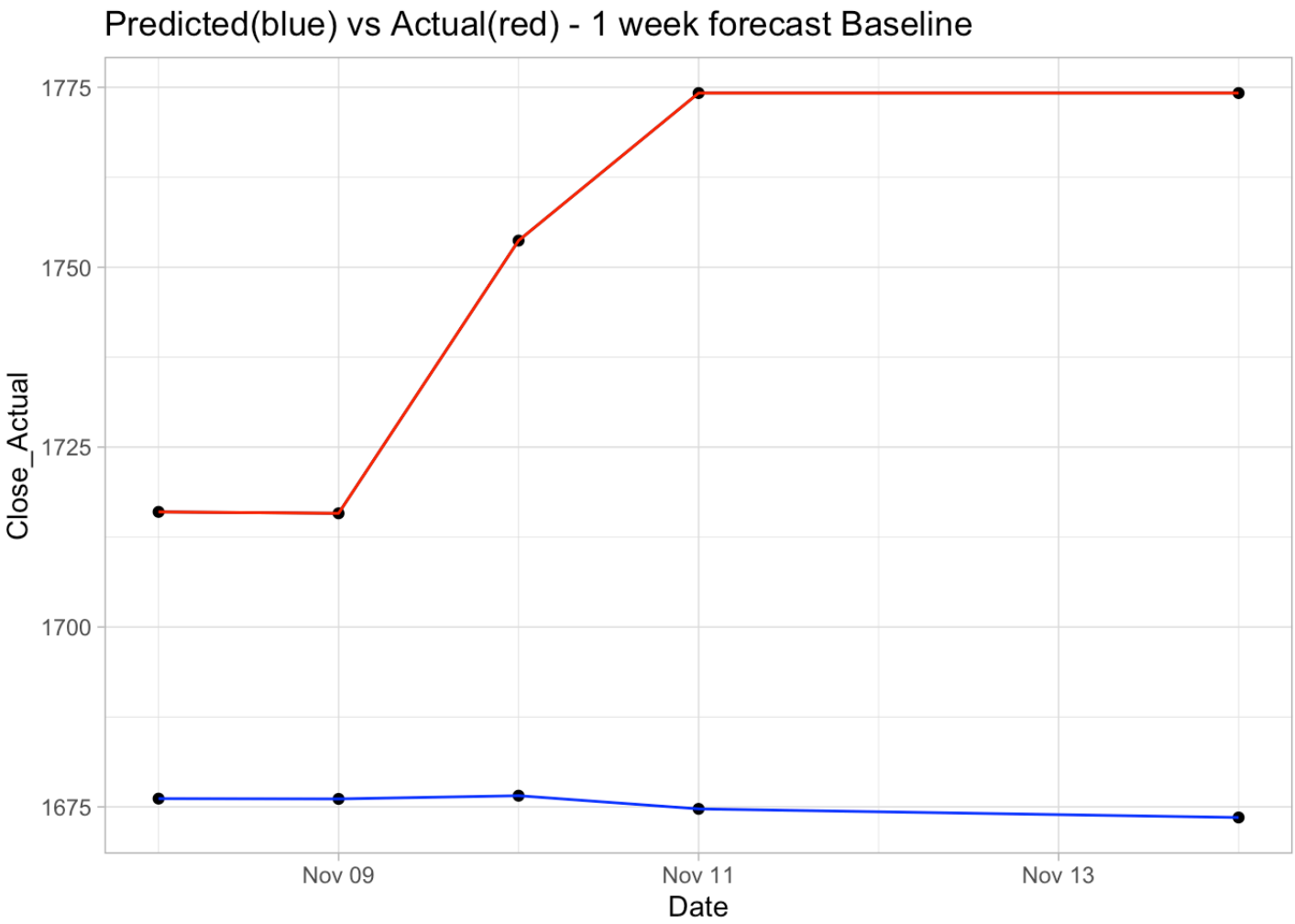
```
## [1] 76.34688
```

Plot Baseline Prophet Forecast

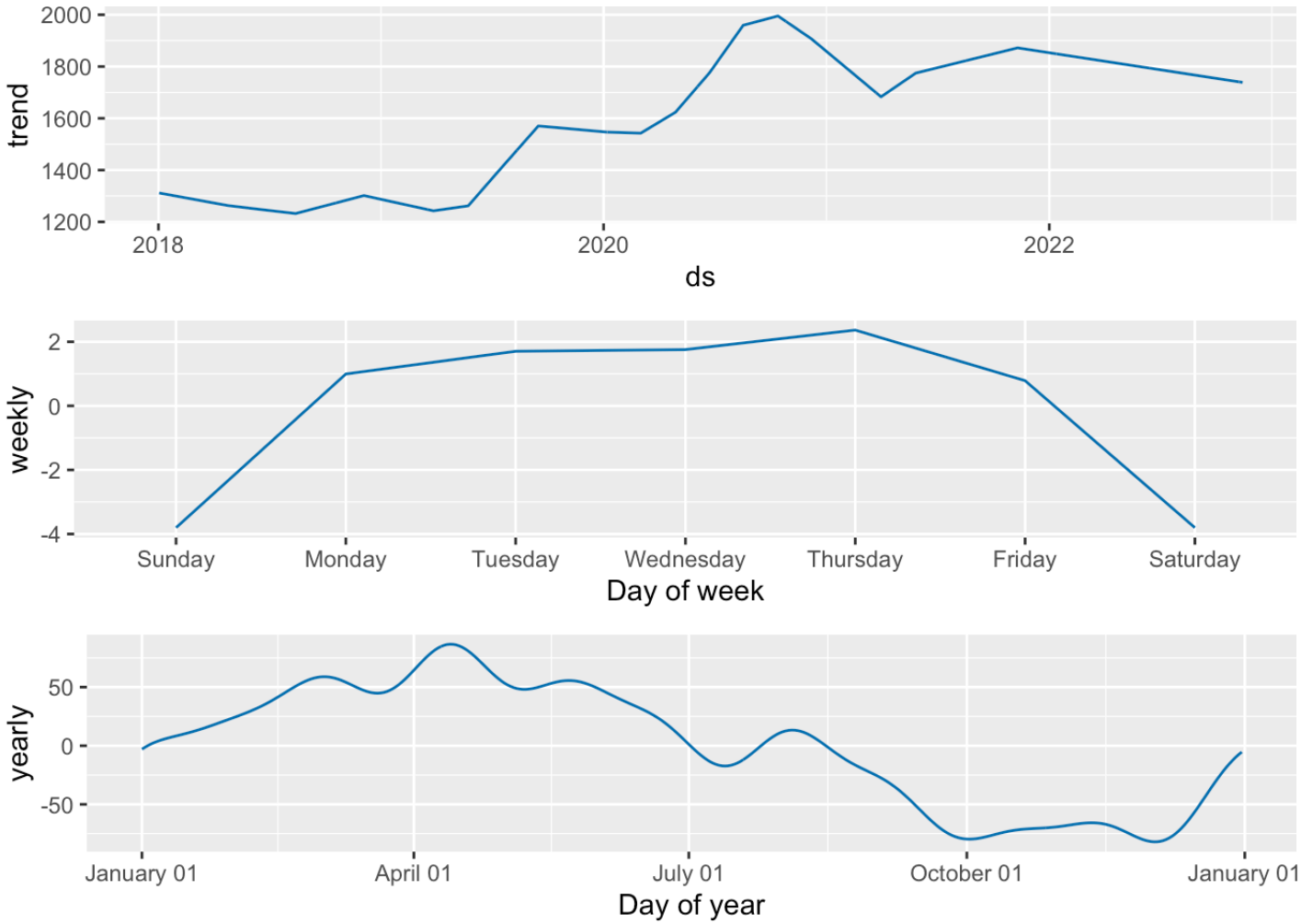
```
# Plot prophet forecast
plot(Prophet, Prophet_Forecast_base, xlabel = "Date", ylabel = "Gold Close Price ($)"
) + ggtitle(paste0("Gold", ": Baseline Price Prediction"))
```




```
# Plot actual
p1 <- ggplot(results, aes(Date, Close_Actual, group=1)) +
    geom_line() +
    theme_light() + ggtitle("Actual 1 Week  Price")

# Plot Predicted
p2 <- ggplot(results, aes(Date, Close_Prediction, group=1)) +
    geom_line() +
    theme_light() + ggtitle("Predicted 1 Week Gold Price")
# Combine Predicted and Actual Plot
p <- p1 +
  geom_point(mapping=p1$mapping) +
  geom_line(color='red') +
  geom_point(mapping=p2$mapping)+
  geom_line(mapping=p2$mapping, color='blue') +
  ggtitle("Predicted(blue) vs Actual(red) - 1 week forecast Baseline")
p
```




```
# Plot prophet components
prophet_plot_components(Prophet, Prophet_Forecast_base)
```



Buy/Sell Baseline Prophet Forecast

```
results$Index <- seq(1, nrow(results), by=1)
vec <- results$Close_Prediction
x <- locate_xtrem(vec)
```

```
## Warning: Use of .data in tidyselect expressions was deprecated in tidyselect 1.2.0
.
## ℹ Please use `"Idx"` instead of `.data$Idx`
```

```
## Warning: Use of .data in tidyselect expressions was deprecated in tidyselect 1.2.0
.
## ℹ Please use `"NB"` instead of `.data$NB`
```

```
x <- x %>%
  select(c('Idx', 'Status')) %>%
  rename(Index = Idx)
results <- left_join(results, x,  by="Index")
results <- results %>%
  select(c('Date', 'Close_Prediction', 'Close_Actual', 'Status'))
results["Status"][results["Status"] == "min"] <- "Buy"
results["Status"][results["Status"] == "max"] <- "Sell"
Profit <- na.omit(results)
Profit <- Profit %>%
  mutate(price_diff = Close_Actual - lag(Close_Actual, default = first(Close_Actual))
) %>%
  filter(Status == 'Sell')
sum(Profit$price_diff) # TWEAK OR USE ONLINE CALC
```

```
## [1] 37.9
```

GOLD FORECAST WEEKLY ROLLOVER

Idea is that if we want to forecast a week ahead, we can use last weeks external values (5 day lag) to forecast gold prices. All data will be known, but continuous forecast require weekly updates. The goal is to predict 1 week in advance, then training data gets expanded 1 week later with actual gold values, retrains model with actual data, and forecasts with 1 week lag predictors.

```
# Create a new full data frame
FULL_df1 <- subset(FULL_df, Date >= as.Date("2018-01-08"))

# Lag External Regressors by 5 (1 week lag)
FULL_df1$Silver_Close.x <- lag(FULL_df1$Silver_Close.x, n=5, default = NA)
FULL_df1$DFF.x <- lag(FULL_df1$DFF.x, n=5, default = NA)
FULL_df1$WM2NS.x <- lag(FULL_df1$WM2NS.x, n=5, default = NA)
FULL_df1$hits.x <- lag(FULL_df1$hits.x, n=5, default = NA)
FULL_df1$DJIA.x <- lag(FULL_df1$DJIA.x, n=5, default = NA)
# Grab data frame to start at the beginning of the week (monday)
FULL_df1 <- subset(FULL_df1, Date >= as.Date("2018-02-12"))

# Store future dataframe (for prophet) with 1 week lag predictors
df_future <- FULL_df1 %>%
  select(c('Date', 'Silver_Close.x', "DFF.x", "WM2NS.x" , "hits.x", "DJIA.x")) %>%
  rename(ds = Date)
```

Gold 1 week rollover prophet forecast

```
# Partition data
prophet.Train <- FULL_df1[1:1160,]
prophet.Test <- FULL_df1[1161:1164,]
```

```
Results <- data.frame()
pred <- data.frame()
# Jumps 4 days in advance
stepsAhead <- 4
# Predicting 4 weeks of forecasts
periods_forecast <- 4
length <- seq(4, stepsAhead * periods_forecast , by=stepsAhead)

# Actual Gold Values
Gold_Results <- gold %>%
  select(c("Date","Gold_Close"))  %>%
  rename(Close_Actual = Gold_Close)

# Create a for loop to retrain model with a new training set
for(i in length) {
  df3 <- prophet.Train %>%
  select(c('Date', 'Gold_Close.x', 'Silver_Close.x', "DFF.x", "WM2NS.x" , "hits.x", "
DJIA.x")) %>%
  rename(ds = Date, y = Gold_Close.x)
  # Add Regressors
  m_ext <- prophet(seasonality.mode = "multiplicative", daily.seasonality = FALSE, in
terval.width = .90)
  m_ext <- add_regressor(m_ext,  "Silver_Close.x", mode = 'multiplicative', standardi
ze = "auto")
  m_ext <- add_regressor(m_ext,  'DFF.x', mode = 'multiplicative', standardize = "aut
o")
  m_ext <- add_regressor(m_ext,  'WM2NS.x', mode = 'multiplicative', standardize = "a
uto")
  m_ext <- add_regressor(m_ext,  'hits.x', mode = 'multiplicative', standardize = "au
to")
  m_ext <- add_regressor(m_ext,  'DJIA.x', mode = 'multiplicative', standardize = "au
to")
  # Fit Model
  m_ext <- fit.prophet(m_ext, df3)
  # Create future dataframe
  future <- make_future_dataframe(m_ext, periods = 6, include_history = TRUE) %>% fil
ter(!wday(ds) %in% c(1,7))
  future <- left_join(future, df_future, by="ds")
  # Forecast with model
  forecast_weekly <- predict(m_ext, future)
  # Return Results and organize data
  Prophet_Results <- forecast_weekly %>%
  select(c("ds","yhat")) %>%
  rename(Date = ds, Close_Prediction = yhat)
  Prophet_Results <- Prophet_Results[(nrow(prophet.Train)+1:(nrow(prophet.Train)+step
sAhead + 1)),]
  Prophet_Results <- na.omit(Prophet_Results)
  pred <- rbind(pred, Prophet_Results )
  # Re-size training and test data
  prophet.Train <- FULL_df1[1:(1160+i),]
```

```
  prophet.Test <- FULL_df1[(1161+i):(1161+i+ stepsAhead),]

}
# Join Predicted and Actual Results
all_results <- left_join(pred, Gold_Results,  by="Date")
all_results
```

Date <dttm>	Close_Prediction <dbl>	Close_Actual <dbl>
2022-10-24	1670.945	1654.1
2022-10-25	1671.979	1658.0
2022-10-26	1669.986	1669.2
2022-10-27	1671.842	1668.8
2022-10-28	1659.312	1648.3
2022-10-31	1668.884	1648.3
2022-11-01	1667.862	1636.4
2022-11-02	1669.719	1651.0
2022-11-03	1656.015	1637.7
2022-11-04	1650.569	1685.7

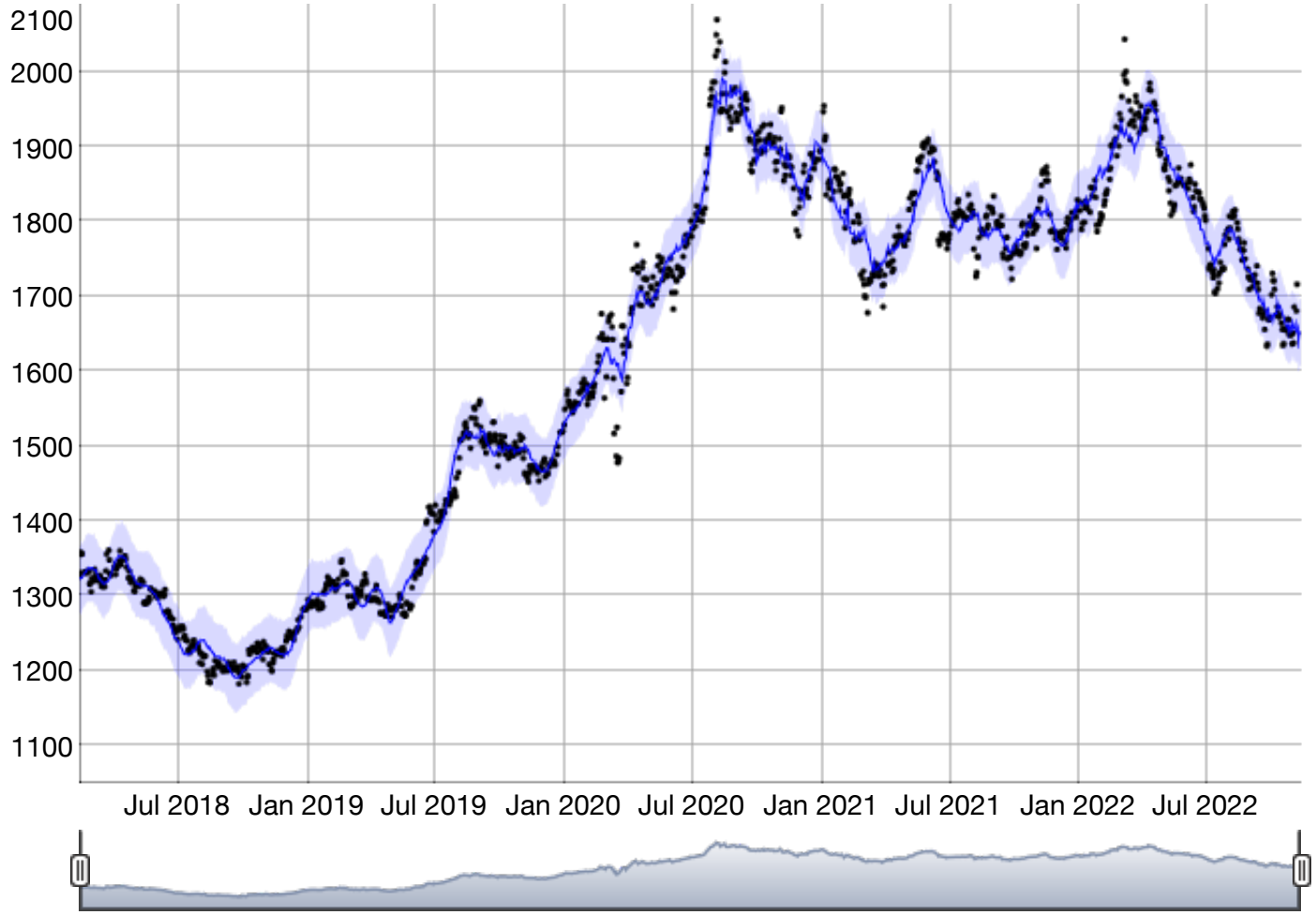

```
# Print RMSE
RMSE <- sqrt(mean((all_results$Close_Actual - all_results$Close_Prediction)^2))
RMSE
```

```
## [1] 58.6095
```

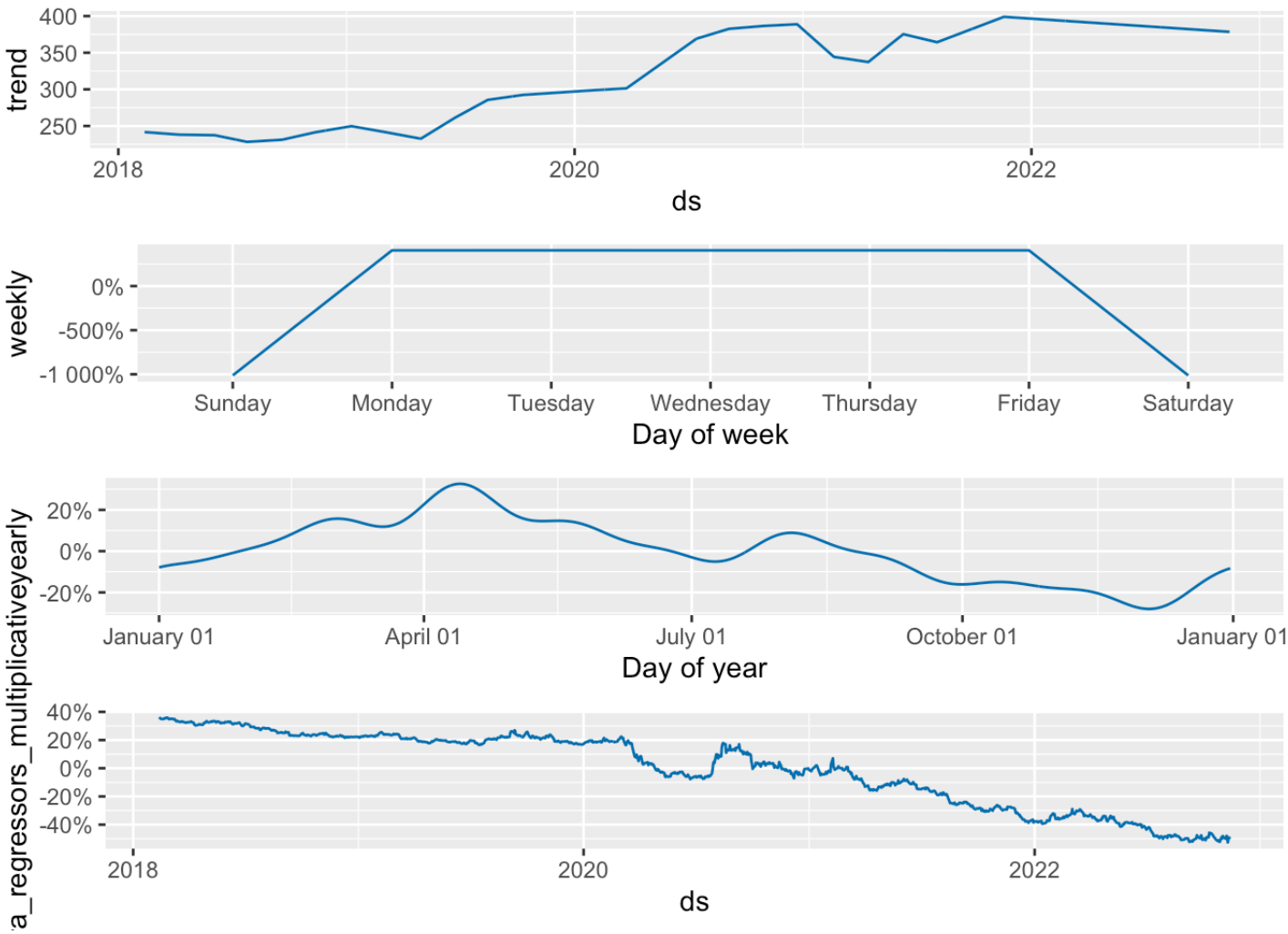
Plot Forecast Values

```
dyplot.prophet(m_ext, forecast_weekly)
```

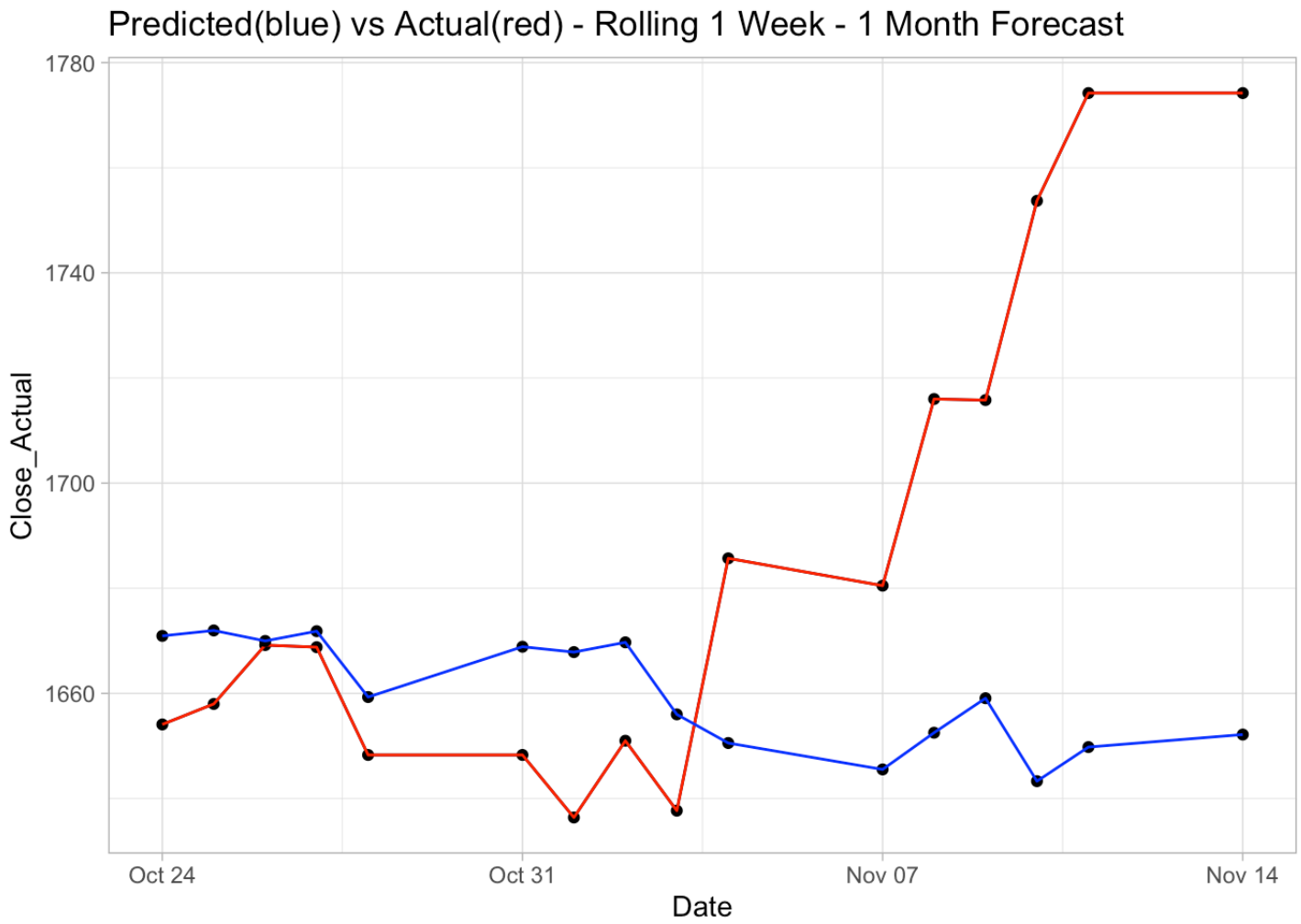
```
## Warning: `select_()` was deprecated in dplyr 0.7.0.
## ℹ Please use `select()` instead.
## ℹ The deprecated feature was likely used in the dplyr package.
##   Please report the issue at <]8;;https://github.com/tidyverse/dplyr/issueshttps://github.com/tidyverse/dplyr/issues]8;;>.
```




```
prophet_plot_components(m_ext, forecast_weekly)
```




```
p1_ext <- ggplot(all_results, aes(Date, Close_Actual, group=1)) +
   geom_line() +
   theme_light() + ggtitle("Actual 1 Week  Price")


 p2_ext <- ggplot(all_results, aes(Date, Close_Prediction, group=1)) +
   geom_line() +
   theme_light() + ggtitle("Predicted 1 Week Gold Price")

 p_ext <- p1_ext +
   geom_point(mapping=p1_ext$mapping) +
   geom_line(color='red') +
   geom_point(mapping=p2_ext$mapping)+
   geom_line(mapping=p2_ext$mapping, color='blue') +
   ggtitle("Predicted(blue) vs Actual(red) - Rolling 1 Week - 1 Month Forecast")
 p_ext
```



Buy/Sell Signal

```
# Find local min and max to create buy/sell signals
all_results$Index <- seq(1, nrow(all_results), by=1)
vec <- all_results$Close_Prediction
x <- locate_xtrem(vec)
x <- x %>%
  select(c('Idx', 'Status')) %>%
  rename(Index = Idx)
all_results <- left_join(all_results, x,  by="Index")
all_results <- all_results %>%
  select(c('Date', 'Close_Prediction', 'Close_Actual', 'Status'))
all_results["Status"][all_results["Status"] == "min"] <- "Buy"
all_results["Status"][all_results["Status"] == "max"] <- "Sell"
Profit <- na.omit(all_results)

# Calculate profit based on buy/sell signals
Profit <- Profit %>%
  mutate(price_diff = Close_Actual - lag(Close_Actual, default = first(Close_Actual))
) %>%
  filter(Status == 'Sell')

sum(Profit$price_diff)  # TWEAK OR USE ONLINE CALC
```

```
## [1] 73.9
```

GOLD FORECAST ROLLOVER 1 Day

Idea is that if we want to forecast 1 day ahead, we can use last weeks external values (1 day lag) to forecast gold prices. All data will be known, but continuous forecast require daily updates. The goal is to predict 1 day in advance, then training data gets expanded 1 day later with actual gold values, retrains model with actual data, and forecasts with 1 day lag predictors.

```
# New dataframe with lagged predictors
FULL_df2 <- subset(FULL_df, Date >= as.Date("2018-01-08"))

# Lag External Regressors by 1 day
FULL_df2$Silver_Close.x <- lag(FULL_df2$Silver_Close.x, n=1, default = NA)
FULL_df2$DFF.x <- lag(FULL_df2$DFF.x, n=1, default = NA)
FULL_df2$WM2NS.x <- lag(FULL_df2$WM2NS.x, n=1, default = NA)
FULL_df2$hits.x <- lag(FULL_df2$hits.x, n=1, default = NA)
FULL_df2$DJIA.x <- lag(FULL_df2$DJIA.x, n=1, default = NA)
FULL_df2 <- subset(FULL_df2, Date >= as.Date("2018-02-12"))
# Store lagged variables in future dataframe for prophet
df_future <- FULL_df2 %>%
  select(c('Date', 'Silver_Close.x', "DFF.x", "WM2NS.x" , "hits.x", "DJIA.x")) %>%
  rename(ds = Date)
```

```
# Partition data
```

```
prophet.Train <- FULL_df2[1:1160,]
prophet.Test <- FULL_df2[1161:1161,]
Results <- data.frame()
pred <- data.frame()
# Jump 1 day in advance
stepsAhead <- 1
# Forecasting for 20 periods aka 20 days
periods_forecast <- 20
length <- seq(1, stepsAhead * periods_forecast , by=stepsAhead)

# Actual gold values
Gold_Results <- gold %>%
  select(c("Date","Gold_Close"))  %>%
  rename(Close_Actual = Gold_Close)

# Create for loop to retrain model with a new training set
for(i in length) {
  df3 <- prophet.Train %>%
  select(c('Date', 'Gold_Close.x', 'Silver_Close.x', "DFF.x", "WM2NS.x" , "hits.x", "
DJIA.x")) %>%
  rename(ds = Date, y = Gold_Close.x)
  # Add regressor
  m_ext <- prophet(seasonality.mode = "multiplicative", daily.seasonality = FALSE, in
terval.width = .90)
  m_ext <- add_regressor(m_ext,  "Silver_Close.x", mode = 'multiplicative', standardi
ze = "auto")
  m_ext <- add_regressor(m_ext,  'DFF.x', mode = 'multiplicative', standardize = "aut
o")
  m_ext <- add_regressor(m_ext,  'WM2NS.x', mode = 'multiplicative', standardize = "a
uto")
  m_ext <- add_regressor(m_ext,  'hits.x', mode = 'multiplicative', standardize = "au
to")
  m_ext <- add_regressor(m_ext,  'DJIA.x', mode = 'multiplicative', standardize = "au
to")
  #Fit Model
  m_ext <- fit.prophet(m_ext, df3)
  # Create future dataframe
  future <- make_future_dataframe(m_ext, periods = 1, include_history = TRUE) %>% fil
ter(!wday(ds) %in% c(1,7))
  future <- left_join(future, df_future, by="ds")
  # Forecast with model
  forecast_weekly <- predict(m_ext, future)

  # Return Results
  Prophet_Results <- forecast_weekly %>%
  select(c("ds","yhat")) %>%
  rename(Date = ds, Close_Prediction = yhat)
  Prophet_Results <- Prophet_Results[(nrow(prophet.Train)+1:(nrow(prophet.Train)+step
sAhead + 1)),]
  Prophet_Results <- na.omit(Prophet_Results)
```

```
  pred <- rbind(pred, Prophet_Results )
  # Resize training data
  prophet.Train <- FULL_df2[1:(1160+i),]
  prophet.Test <- FULL_df2[(1161+i):(1161+i+ stepsAhead),]

}

# Join Predicted and actual Results
all_results <- left_join(pred, Gold_Results,  by="Date")
all_results
```

Date <dttm>	Close_Prediction <dbl>	Close_Actual <dbl>
2022-10-25	1670.467	1658.0
2022-10-26	1657.114	1669.2
2022-10-27	1674.551	1668.8
2022-10-28	1665.178	1648.3
2022-11-01	1652.849	1636.4
2022-11-02	1666.021	1651.0
2022-11-03	1662.231	1637.7
2022-11-04	1613.514	1685.7
2022-11-08	1672.081	1716.0
2022-11-09	1677.143	1715.8

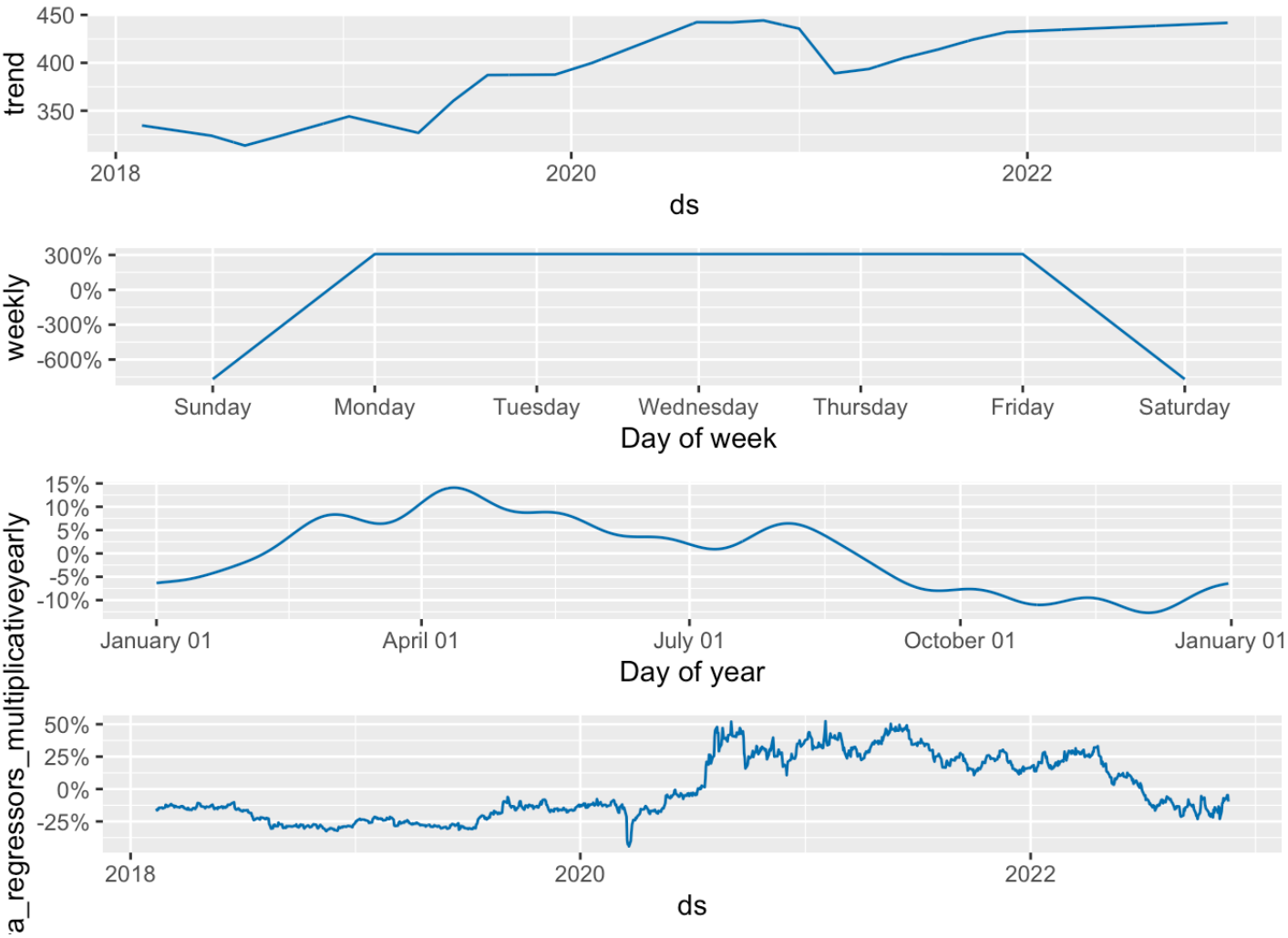

```
# RMse
RMSE <- sqrt(mean((all_results$Close_Actual - all_results$Close_Prediction)^2))
RMSE
```

```
## [1] 43.45533
```

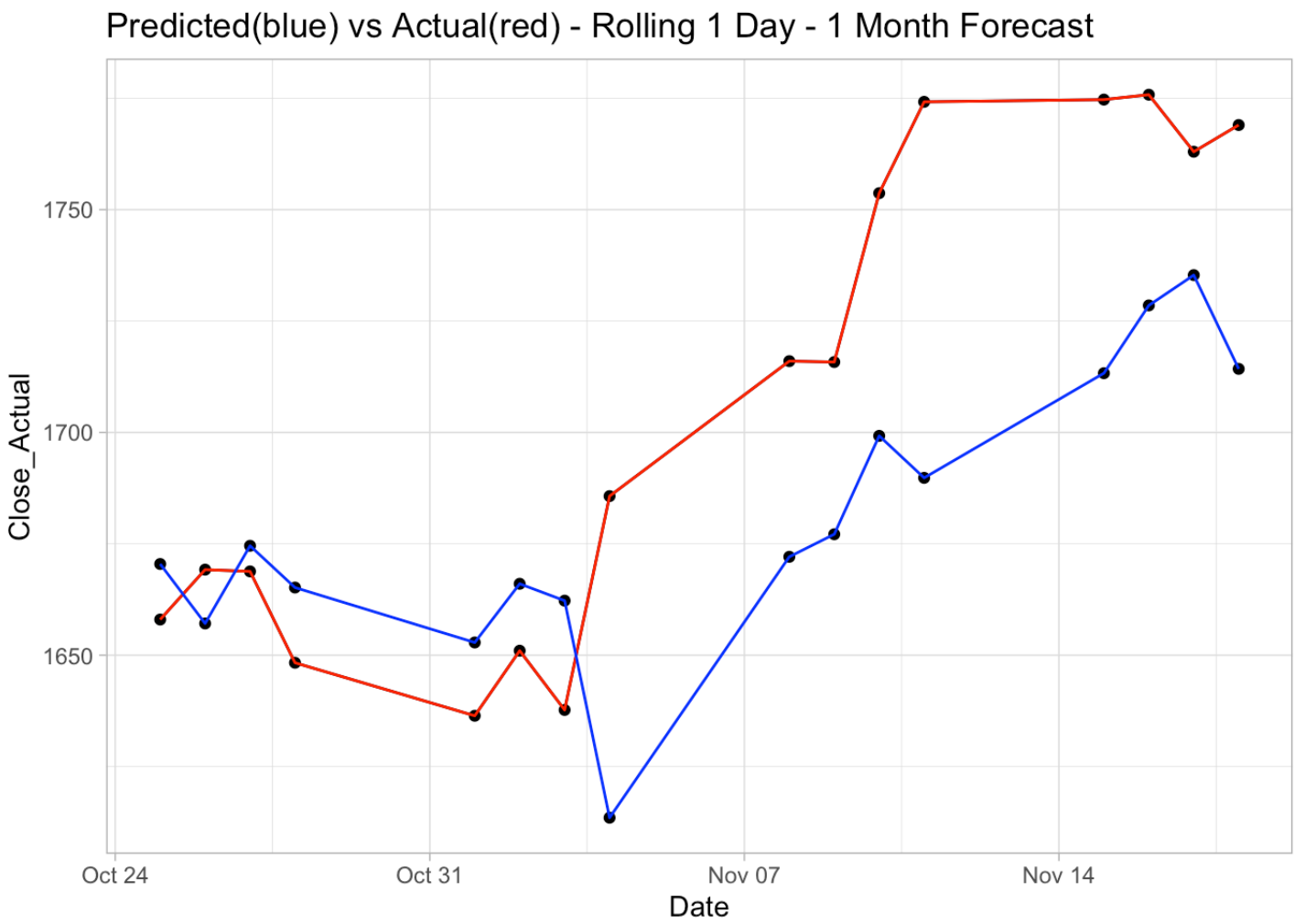
Plot Forecast

```
# FOrecast plot
dyplot.prophet(m_ext, forecast_weekly)
```


```
# Plot components
prophet_plot_components(m_ext, forecast_weekly)
```




```
# Plot actual and predicted data
 p1_ext <- ggplot(all_results, aes(Date, Close_Actual, group=1)) +
    geom_line() +
    theme_light() + ggtitle("Actual 1 Week  Price")

 p2_ext <- ggplot(all_results, aes(Date, Close_Prediction, group=1)) +
    geom_line() +
    theme_light() + ggtitle("Predicted 1 Week Gold Price")

p_ext <- p1_ext +
    geom_point(mapping=p1_ext$mapping) +
    geom_line(color='red') +
    geom_point(mapping=p2_ext$mapping)+
    geom_line(mapping=p2_ext$mapping, color='blue') +
    ggtitle("Predicted(blue) vs Actual(red) - Rolling 1 Day - 1 Month Forecast")
 p_ext
```



Buy/Sell Signal

```
all_results$Index <- seq(1, nrow(all_results), by=1)
vec <- all_results$Close_Prediction
x <- locate_xtrem(vec)
x <- x %>%
  select(c('Idx', 'Status')) %>%
  rename(Index = Idx)
all_results <- left_join(all_results, x,  by="Index")
all_results <- all_results %>%
  select(c('Date', 'Close_Prediction', 'Close_Actual', 'Status'))
all_results["Status"][all_results["Status"] == "min"] <- "Buy"
all_results["Status"][all_results["Status"] == "max"] <- "Sell"
Profit <- na.omit(all_results)
Profit <- Profit %>%
  mutate(price_diff = Close_Actual - lag(Close_Actual, default = first(Close_Actual))
) %>%
  filter(Status == 'Sell')
sum(Profit$price_diff) # TWEAK OR USE ONLINE CALC
```

```
## [1] 71
```

No Lag on Predictors (Check to see if external regressors help prediction if forecasted perfectly)

```
# New dataframe with lagged predictors
FULL_df2 <- subset(FULL_df, Date >= as.Date("2018-01-08"))

# Lag External Regressors by 1 day
FULL_df2$Silver_Close.x <- FULL_df2$Silver_Close.x
FULL_df2$DFF.x <- FULL_df2$DFF.x
FULL_df2$WM2NS.x <- FULL_df2$WM2NS.x
FULL_df2$hits.x <- FULL_df2$hits.x
FULL_df2$DJIA.x <- FULL_df2$DJIA.x
FULL_df2 <- subset(FULL_df2, Date >= as.Date("2018-02-12"))
# Store lagged variables in future dataframe for prophet
df_future <- FULL_df2 %>%
  select(c('Date', 'Silver_Close.x', "DFF.x", "WM2NS.x" , "hits.x", "DJIA.x")) %>%
  rename(ds = Date)
```

```
# Partition data
prophet.Train <- FULL_df2[1:1160,]
prophet.Test <- FULL_df2[1161:1161,]
Results <- data.frame()
pred <- data.frame()
# Jump 1 day in advance
stepsAhead <- 1
# Forecasting for 20 periods aka 20 days
periods_forecast <- 20
length <- seq(1, stepsAhead * periods_forecast , by=stepsAhead)

# Actual gold values
Gold_Results <- gold %>%
  select(c("Date","Gold_Close"))  %>%
  rename(Close_Actual = Gold_Close)

# Create for loop to retrain model with a new training set
for(i in length) {
  df3 <- prophet.Train %>%
  select(c('Date', 'Gold_Close.x', 'Silver_Close.x', "DFF.x", "WM2NS.x" , "hits.x", "
DJIA.x")) %>%
  rename(ds = Date, y = Gold_Close.x)
  # Add regressor
  m_ext <- prophet(seasonality.mode = "multiplicative", daily.seasonality = FALSE, in
terval.width = .90)
  m_ext <- add_regressor(m_ext,  "Silver_Close.x", mode = 'multiplicative', standardi
ze = "auto")
```

```
  m_ext <- add_regressor(m_ext,  'DFF.x', mode = 'multiplicative', standardize = "aut
o")
  m_ext <- add_regressor(m_ext,  'WM2NS.x', mode = 'multiplicative', standardize = "a
uto")
  m_ext <- add_regressor(m_ext,  'hits.x', mode = 'multiplicative', standardize = "au
to")
  m_ext <- add_regressor(m_ext,  'DJIA.x', mode = 'multiplicative', standardize = "au
to")
  #Fit Model
  m_ext <- fit.prophet(m_ext, df3)
  # Create future dataframe
  future <- make_future_dataframe(m_ext, periods = 1, include_history = TRUE) %>% fil
ter(!wday(ds) %in% c(1,7))
  future <- left_join(future, df_future, by="ds")
  # Forecast with model
  forecast_weekly <- predict(m_ext, future)

  # Return Results
  Prophet_Results <- forecast_weekly %>%
  select(c("ds","yhat")) %>%
  rename(Date = ds, Close_Prediction = yhat)
  Prophet_Results <- Prophet_Results[(nrow(prophet.Train)+1:(nrow(prophet.Train)+step
sAhead + 1)),]
  Prophet_Results <- na.omit(Prophet_Results)
  pred <- rbind(pred, Prophet_Results )
  # Resize training data
  prophet.Train <- FULL_df2[1:(1160+i),]
  prophet.Test <- FULL_df2[(1161+i):(1161+i+ stepsAhead),]

}

# Join Predicted and actual Results
all_results <- left_join(pred, Gold_Results,  by="Date")
all_results
```

Date <dttm>	Close_Prediction <dbl>	Close_Actual <dbl>
2022-10-25	1644.479	1658.0
2022-10-26	1667.214	1669.2
2022-10-27	1659.611	1668.8
2022-10-28	1647.537	1648.3
2022-11-01	1664.852	1636.4
2022-11-02	1656.454	1651.0
2022-11-03	1616.411	1637.7

2022-11-04	1639.876	1685.7
2022-11-08	1674.971	1716.0
2022-11-09	1700.381	1715.8

1-10 of 16 rows Previous **1** 2 Next

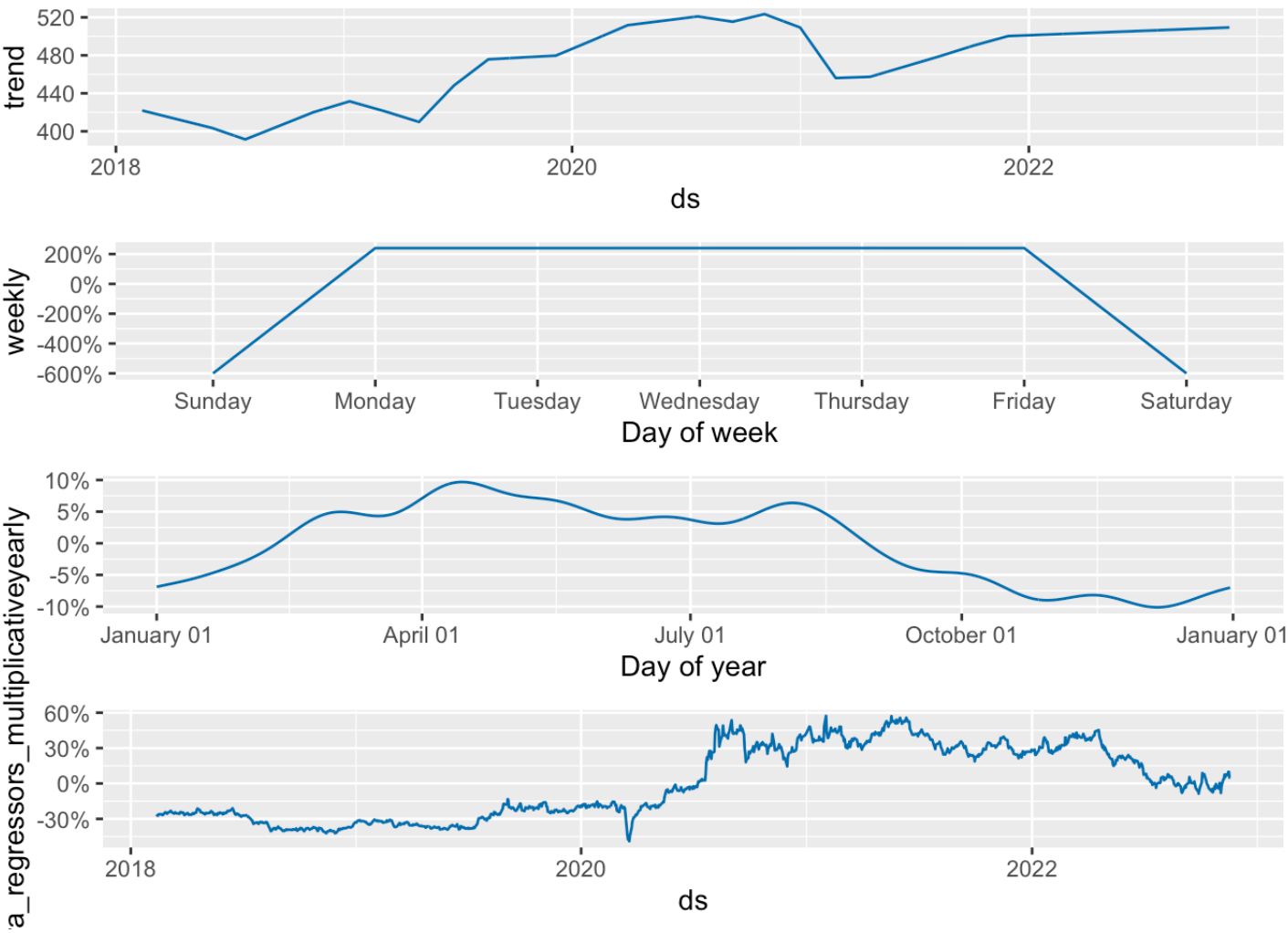
```
# RMse
RMSE <- sqrt(mean((all_results$Close_Actual - all_results$Close_Prediction)^2))
RMSE
```

```
## [1] 39.46699
```

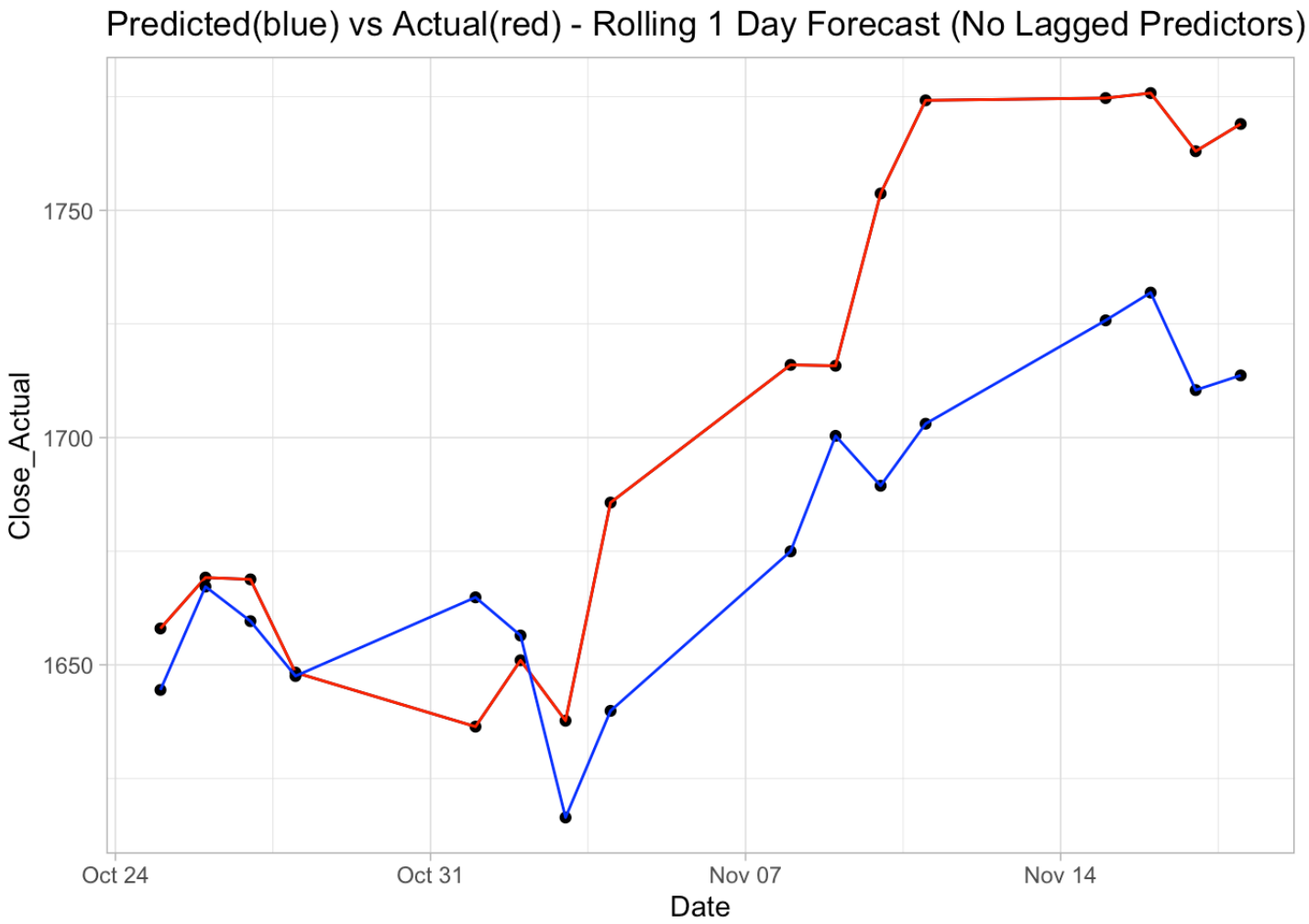
Plot Forecast

```
# FOrecast plot
dyplot.prophet(m_ext, forecast_weekly)
```

```
# Plot components
prophet_plot_components(m_ext, forecast_weekly)
```




```
# Plot actual and predicted data
 p1_ext <- ggplot(all_results, aes(Date, Close_Actual, group=1)) +
    geom_line() +
    theme_light() + ggtitle("Actual 1 Week  Price")

 p2_ext <- ggplot(all_results, aes(Date, Close_Prediction, group=1)) +
    geom_line() +
    theme_light() + ggtitle("Predicted 1 Week Gold Price")

p_ext <- p1_ext +
    geom_point(mapping=p1_ext$mapping) +
    geom_line(color='red') +
    geom_point(mapping=p2_ext$mapping)+
    geom_line(mapping=p2_ext$mapping, color='blue') +
    ggtitle("Predicted(blue) vs Actual(red) - Rolling 1 Day Forecast (No Lagged Predi
ctors)")
 p_ext
```



Buy/Sell Signal

```
all_results$Index <- seq(1, nrow(all_results), by=1)
vec <- all_results$Close_Prediction
x <- locate_xtrem(vec)
x <- x %>%
  select(c('Idx', 'Status')) %>%
  rename(Index = Idx)
all_results <- left_join(all_results, x,  by="Index")
all_results <- all_results %>%
  select(c('Date', 'Close_Prediction', 'Close_Actual', 'Status'))
all_results["Status"][all_results["Status"] == "min"] <- "Buy"
all_results["Status"][all_results["Status"] == "max"] <- "Sell"
Profit <- na.omit(all_results)
Profit <- Profit %>%
  mutate(price_diff = Close_Actual - lag(Close_Actual, default = first(Close_Actual))
) %>%
  filter(Status == 'Sell')
sum(Profit$price_diff) # TWEAK OR USE ONLINE CALC
```

```
## [1] 105.5
```