Comps Report: Predicting Stock Prices

Brandon Wang

4/24/20

Introduction

The stock market provides people with the ability to invest in almost every possible firm. There are endless opportunities for the average person to profit from it. The prices of stocks, however, are unpredictable. The potential gains from the market are not without risks. Large drops in the market, such as during the Great Recession, can lead to financial turmoil for thousands of people. The Recession devastated both the stock market and the entire economy. Many people lost their jobs and homes as a result of the economic downturn. To understand the future of the market, we must look at historical data to forecaststock market trends. The economy itself is heavily tied to the stock market. Understanding the general trend of the market can reveal whether or not the economy as a whole is doing well.

Background

The data I will be looking at is time series data. Time series data is simply data that has been collected over time. Time series analysis uses previously observed values to try and forecase future values (Cryer and Chan, 2008). The first step in my analysis will be to construct an Auto Regressive Integrated Moving Average (ARIMA) model with this data. An ARIMA model is a type of time series model that uses past values of the time series to foreast future values (Hyndman and Athanasopoulos, 2018). A basic ARIMA model contains three different types of terms: autoregressive, integrated, and moving average (Hyndman and Athanasopoulos, 2018). An ARIMA model can be written as:

$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$

Within this model, y'_t represents the time series after it has been differenced (Hyndman and Athanasopoulos, 2018). The ϕ terms represent the autoregressive parts of the model, and the θ terms are the moving average parts (Hyndman and Athanasopoulos, 2018). The ϵ terms are the lagged errors. An ARIMA model can have any number of each type of term (Hyndman and Athanasopoulos, 2018). In these models, the predictor variables would be the lagged y_t values and the lagged errors (Hyndman and Athanasopoulos, 2018). The ϕ and θ terms would be our coefficients. To build one of these models, R just needs to know how many terms to include in the model. The arima() function in R will take in p, d, and q values to indicate the number of autoregressive terms, number of differences, and number of moving average terms respectively.

To build an ARIMA model, our time series must be stationary and non-seasonal (Hyndman and Athanasopoulos, 2018). Once these conditions have been satisfied, we need to figure out how many of each type of term our model needs. The actual model can contain any number of each type of term (Hyndman and Athanasopoulos, 2018).

Autocorrelation measures the linear relationship between lagged variables (Hyndman and Athanasopoulos, 2018). In other words, autocorrelations are how correlated the data is with itself at a different point in time. Stationary data is data that maintains constant mean, variance, and autocorrelations over time (Hyndman and Athanasopoulos, 2018). On the other hand, non-stationary data will vary over time. The Augmented Dickey-Fuller Test and the Kwiatkowski-Phillips-Schmidt-Shin, or KPSS, Test for Level Stationarity can be used to test for stationarity in the data (Hyndman and Athanasopoulos, 2018). The Augmented Dickey-Fuller Test has a null hypothesis that the data is not stationary. The null hypothesis for the KPSS Test, however, is that the data is stationary. Running these tests will tell us if the data are stationary or not. If the data are not stationary, differencing can help transform it to stationary. Differencing involves computing the differences between consecutive observations instead of the observations themselves (Hyndman and Athanasopoulos, 2018). By doing so, we stabilize the mean by removing changes in the level of the time series (Hyndman and Athanasopoulos, 2018).

Seasonal data is data that has a repeating pattern over time (Hyndman and Athanasopoulos, 2018). In other words, the time at which the data is collected would have an effect on the data itself. To test for seasonality, I can use R to decompose the data. R decomposes the data into four separate components: observed, trend, seasonal, and random (Hyndman and Athanasopoulos, 2018). The plots of these components can help detect seasonality. If the data are seasonal, then the seasonality component should show a repeating pattern. This seasonal component can be removed by simply subtracting it out of the data. If the data are not seasonal, then nothing needs to be done.

To determine how many of each term are needed for our model, we need to first create autocorrelation and partial autocorrelation plots. Partial autocorrelations are an extension of autocorrelation where the correlations within the lag are removed (Hyndman and Athanasopoulos, 2018). The patterns in the plots can help guide how many moving average or autoregressive terms are needed. Once we have an idea of how many of these terms to include in the model, we can just test out a few possibilites for the remaining terms and evaluate which of those models is the best.

Once the ARIMA model is built, we can perform intervention analysis on the data if appropriate. Intervention analysis allows us to assess the effect of an intervening event on a time series (Cryer and Chan, 2008). We need a pulse, or specific point in time, to perform intervention analysis. This point in time is when the intervening event occurred. Intervention analysis builds off of an ARIMA model by adding an extra term m_t to address the effects of an intervention (Cryer and Chan, 2008).

The m_t term, which represents the intervention, can be modeled by the following function:

$$m_t = w_1 m_{t-1} + \delta_0 P_t$$

This function is called a transfer function (Cryer and Chan, 2008). For any point before the pulse, this function will simply go to zero. At the pulse, the function will equal δ_0. After the pulse, the function will be equal to $w_1 m_{t-1}$. This equation means that as we move away from the pulse, its effect will decrease. Its effect on each day will be the preceding day's effect times our w_1 coefficient. In other words, there is no effect before the pulse, an immediate effect at the pulse, and a gradually decreasing effect afterwards. The final model will be in the same form as an ARIMA model with the addition of an m_t term.

Data

The data set I found contains daily stock price data from 1998-2019 for over 7,000 different stocks. The data was scraped from several different sources and is hosted on Kaggle. All the data is presented in USD. The variables in the dataset are:

1. open

- stock's start of day price

2. high

- stock's highest price that day

3. low

- stock's lowest price that day

4. close

- stock's end of day price

5. adjusted close

- stock's end of day price adjusted for stock splits

6. volume

- how many times stock was traded that day

```
data <- read_csv("stock_prices_latest.csv")
head(data)
```

```
## # A tibble: 6 x 9
##   symbol date        open  high   low close close_adjusted volume
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>          <dbl>  <dbl>
## 1 MSFT   2016-05-16  50.8  52.0  50.8  51.8           49.7 2.00e7
## 2 MSFT   2002-01-16  68.8  69.8  67.8  67.9           22.6 3.10e7
## 3 MSFT   2001-09-18  53.4  55    53.2  54.3           18.1 4.16e7
## 4 MSFT   2007-10-26  36.0  36.0  34.6  35.0           27.2 2.88e8
## 5 MSFT   2014-06-27  41.6  42.3  41.5  42.2           38.7 7.46e7
## 6 MSFT   2011-11-04  26.4  26.4  26    26.2           22.2 3.65e7
## # ... with 1 more variable: split_coefficient <dbl>
```

Data Wrangling

Although this dataset contained data for over 7,000 stocks, it would take much too long to create an analysis for each different stock. There are tens of millions of daily observations recorded in this dataset. To focus this project, I chose to look at only Bank of America's stock (BAC). Bank of America is a very large, well-known bank, so its prices should hopefully be reflective of the general economy. The first step I took was to filter out all other stocks from the data and then plot BAC's open price for each day.

Figure 1 shows the opening prices for the Bank of America stock for every day from 1998-2018. The graph shows a very sharp sudden decrease right before 2005. This drop is very fast and is unexplained by an economic recession like the later drops are. Upon further investigation, this drop occurred because of a stock split on August 27, 2004 (Wathen, 2018). A stock split occurs when a firm increases the number of its stock available by cutting the price. The firm's actual valuation does not change because it would now have twice as many stocks at half the price. By splitting its stock and cutting the price, a firm can make its stock more accessible to the public. Cheaper stocks can help firms raise capital quickly by making their stock affordable for a wider audience. To account for the stock split, I decided to use adjusted closing prices instead of opening prices for my analysis. The adjusted closing prices have already been adjusted to account for any historical stock splits.

```
ggplot(data=bacdata, aes(x=date, y=close_adjusted)) + geom_line() + labs(y="Adjusted Closing Price", x=
```

As seen in Figure 2, using adjusted closing prices solves the issue of the sudden drop at the stock split date. The graph now shows the stock price steadily increasing all the way until the Great Recession.

ARIMA

In order to actually do any time series analysis, the first thing that needs to be done is converting the dataset into a time series object. This step is necessary because R will treat time series differently from other data. The frequency option lets us choose how many observations are recorded in each cycle of the time series. I set this option to 253 because there are 253 trading days in each year

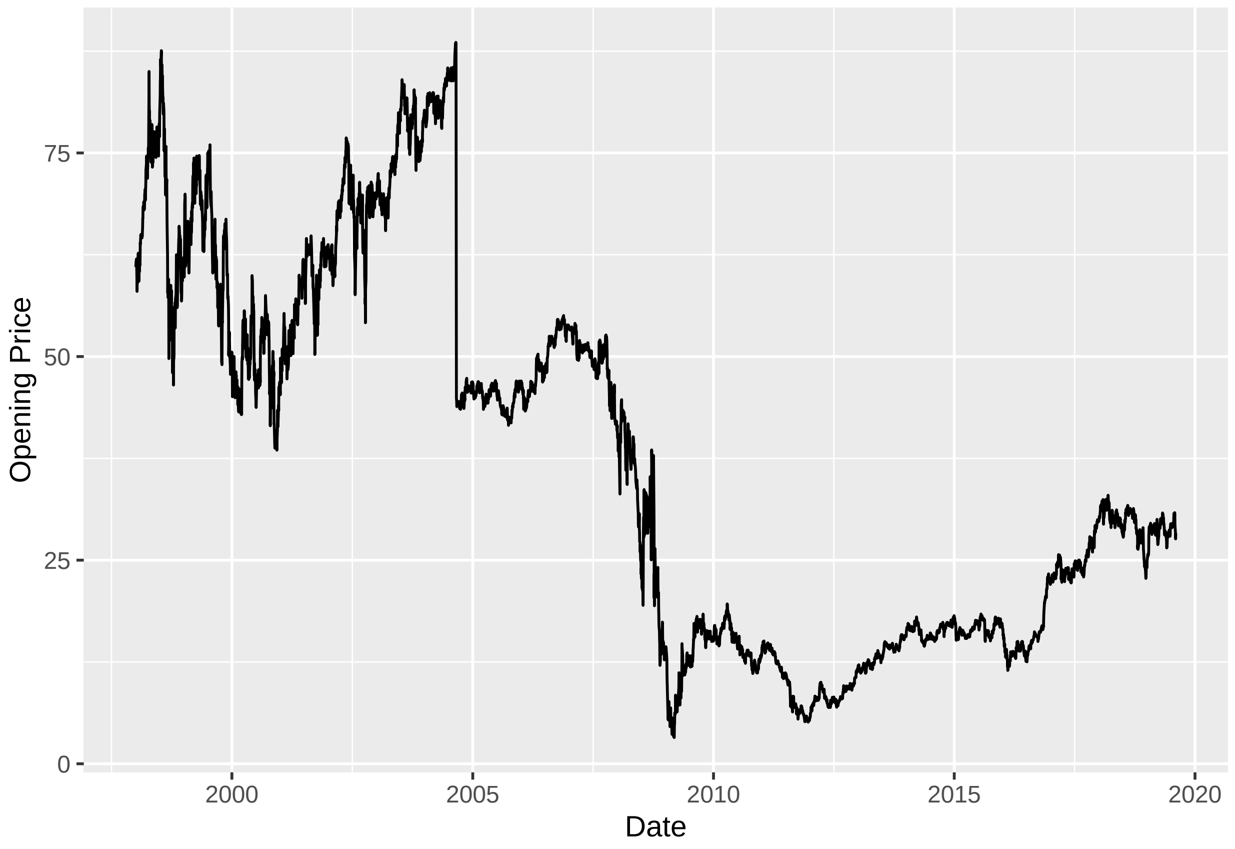


Figure 1: Bank of America daily opening stock price (1998-2019)

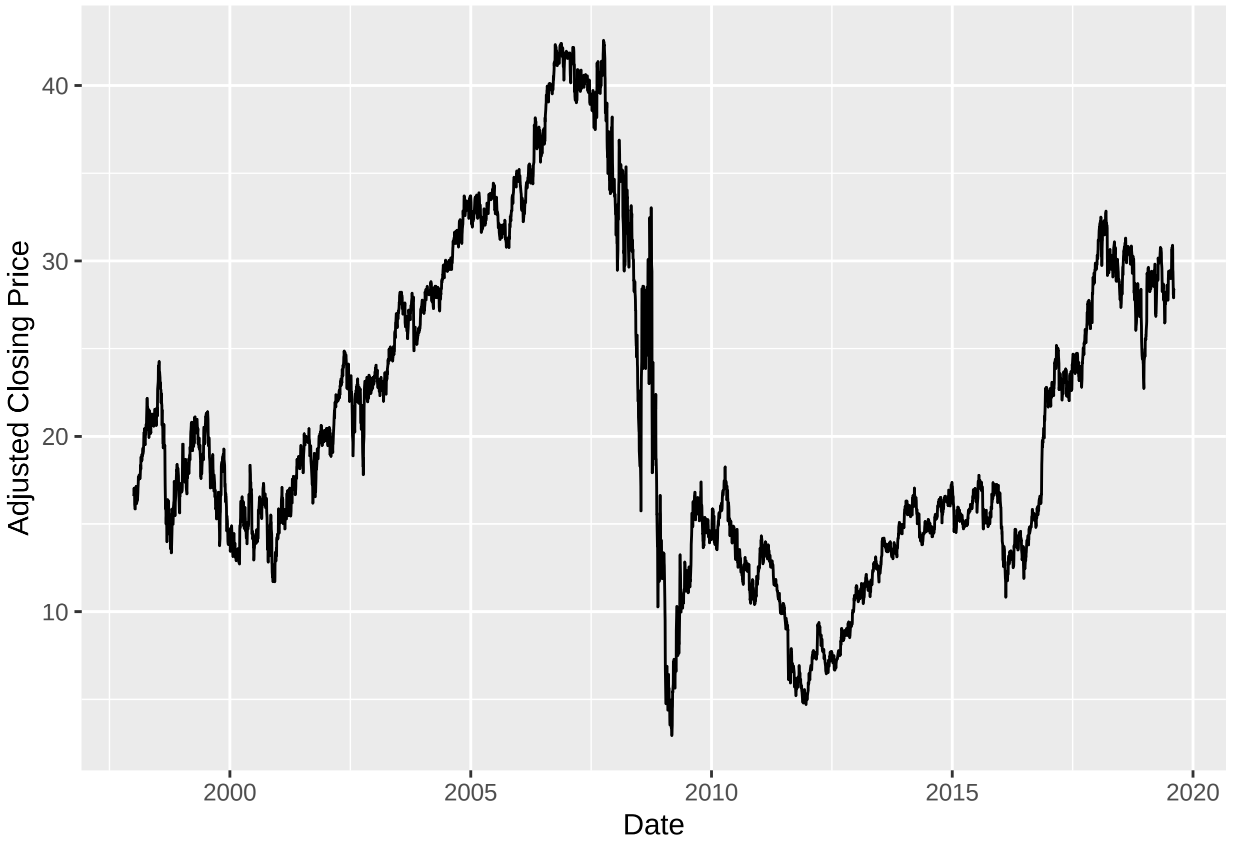


Figure 2: Bank of America daily adjusted closing stock price (1998-2019)

```
ordered <- bacdata[order(as.Date(bacdata$date, format="%Y/%m/%d")),]
tsobj <- ts(ordered$close_adjusted, start = c(1998), frequency = 253)
```

Stationarity

The first step in determining whether or not the data is stationary is simply plotting the data. The data should stay relatively consistent over time if it is stationary. The Augmented Dickey-Fuller Test and KPSS test can then confirm the intuition from looking at the plot.

```
adf.test(tsobj)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  tsobj
## Dickey-Fuller = -1.5403, Lag order = 17, p-value = 0.773
## alternative hypothesis: stationary
```

```
kpss.test(tsobj)
```

```
## Warning in kpss.test(tsobj): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsobj
## KPSS Level = 5.9449, Truncation lag parameter = 10, p-value = 0.01
```

Looking at Figure 2, it appears that the data is not stationary. The mean and variation of the data definitely vary over time. The mean is higher earlier in time, while the variation is lower. As time progresses, the mean decreases and the variation increases. The results of the first Augmented Dickey-Fuller Test and KPSS Test confirm that the data are not stationary. The large p-value from the Augmented Dickey-Fuller Test and the small one from the KPSS test both would suggest that the data are not stationary.

```
diff1 <- diff(tsobj)
autoplot(diff1) + labs(y="Adjusted Closing Price", x="Date")
```

```
adf.test(diff1)
```

```
## Warning in adf.test(diff1): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff1
## Dickey-Fuller = -18.479, Lag order = 17, p-value = 0.01
## alternative hypothesis: stationary
```

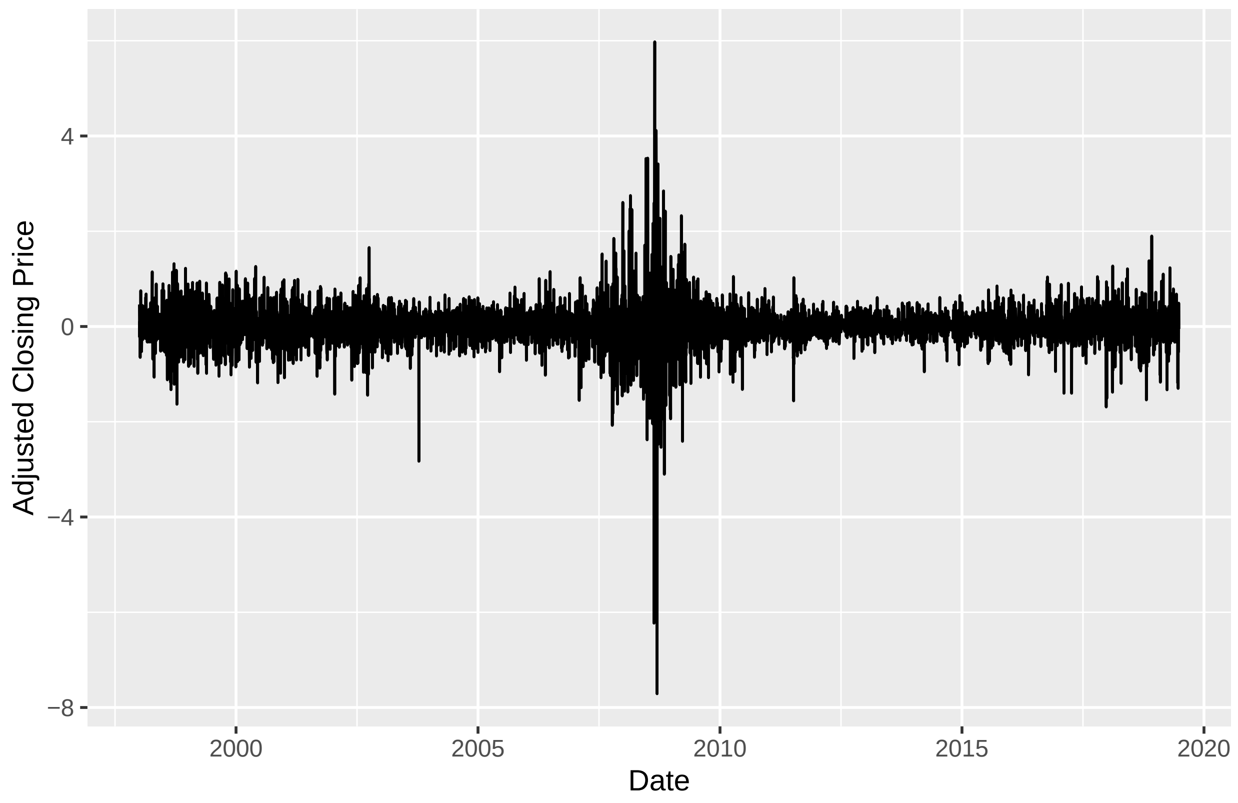


Figure 3: Differences of Bank of America daily adjusted closing stock price (1998-2019

```r
kpss.test(diff1)
```

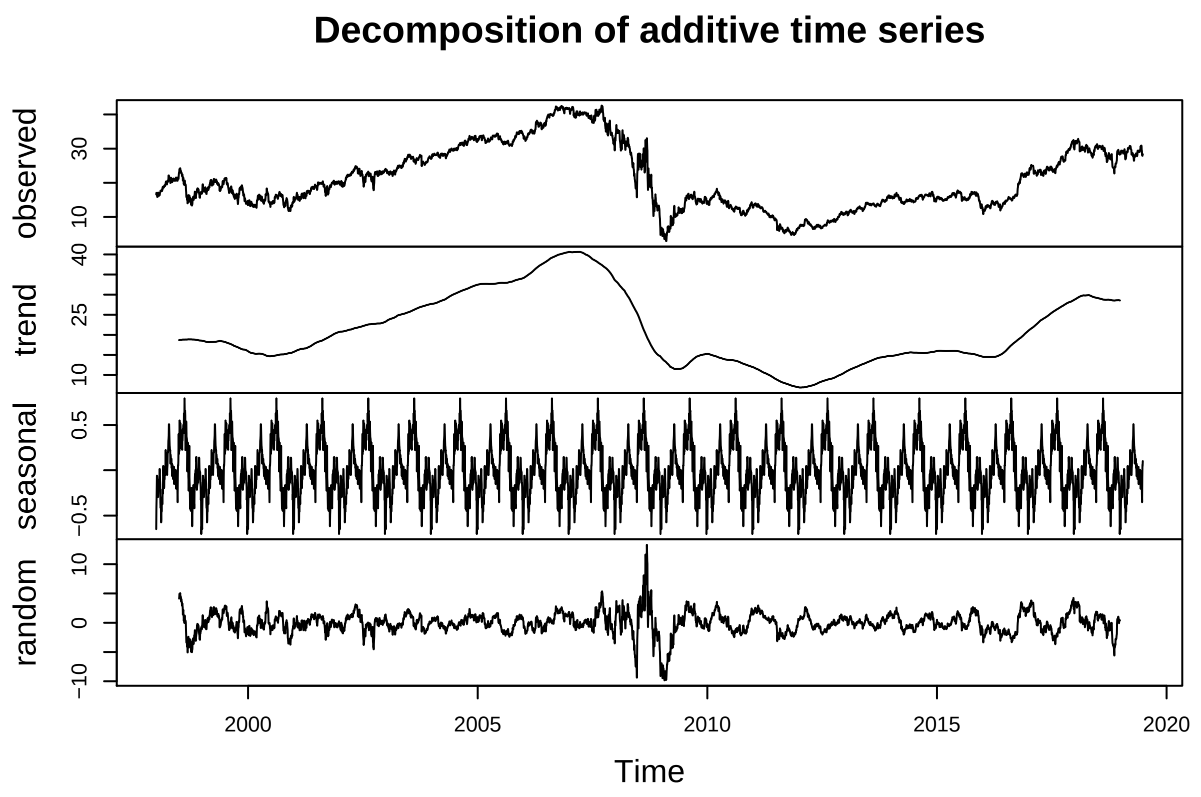
```
## Warning in kpss.test(diff1): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  diff1
## KPSS Level = 0.10135, Truncation lag parameter = 10, p-value = 0.1
```

In order to make the data stationary, I used a technique called differencing. Figure 3 shows the data after it has been differenced. After differencing once, the data appeared to become much more stationary. The Augmented Dickey-Fuller Test and KPSS test both agreed that the differenced data is stationary. The second Augmented Dickey-Fuller Test above shows a low p-value and the KPSS test shows a high one. Both of these results would lead to the conclusion that there is significant evidence the dataset is stationary.

Seasonality

```r
decomp <- decompose(tsobj)
plot(decomp)
```



From the plot of the time series components, it appears that this time series might be seasonal. The seasonal component has a consistent repeating pattern across the entire time series. I will subtract off the seasonal componenet and see if that makes the data nonseasonal.

```r
newts <- tsobj - decomp$seasonal
autoplot(newts) + labs(y="Adjusted Closing Price", x="Date")
```

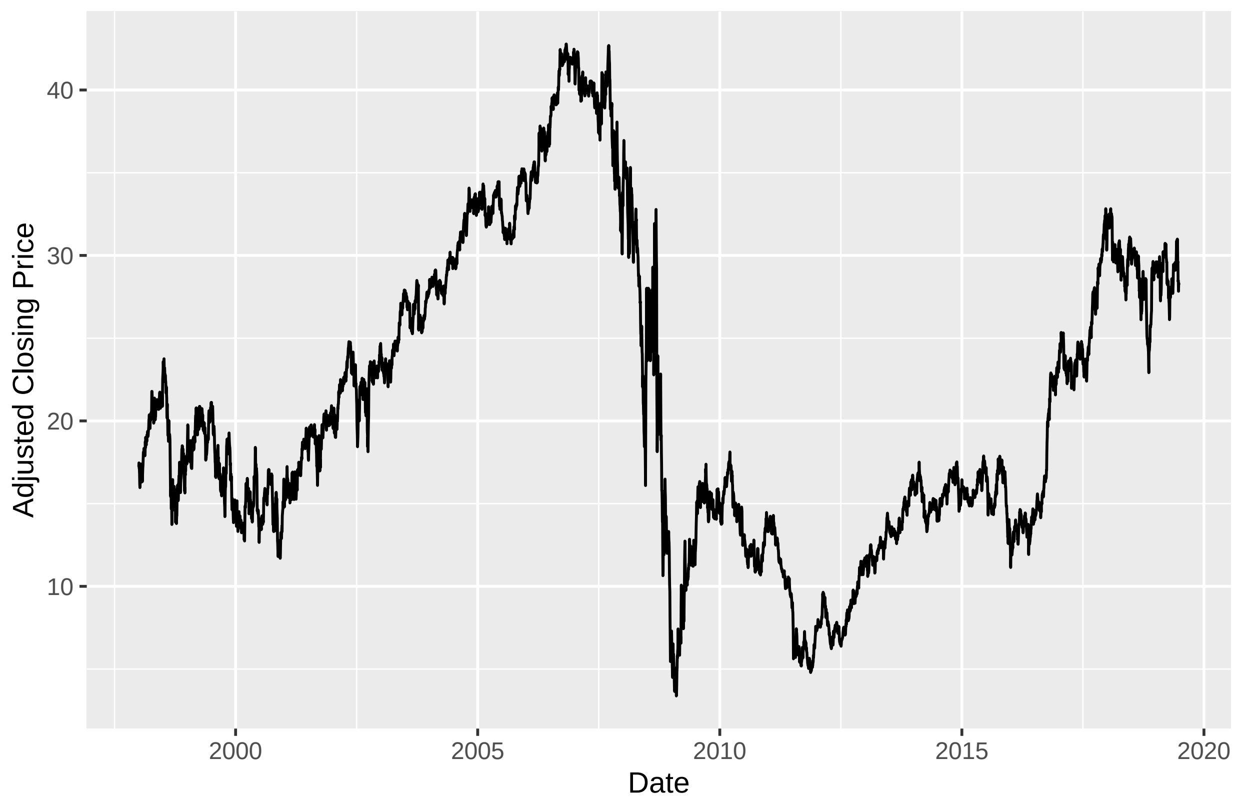


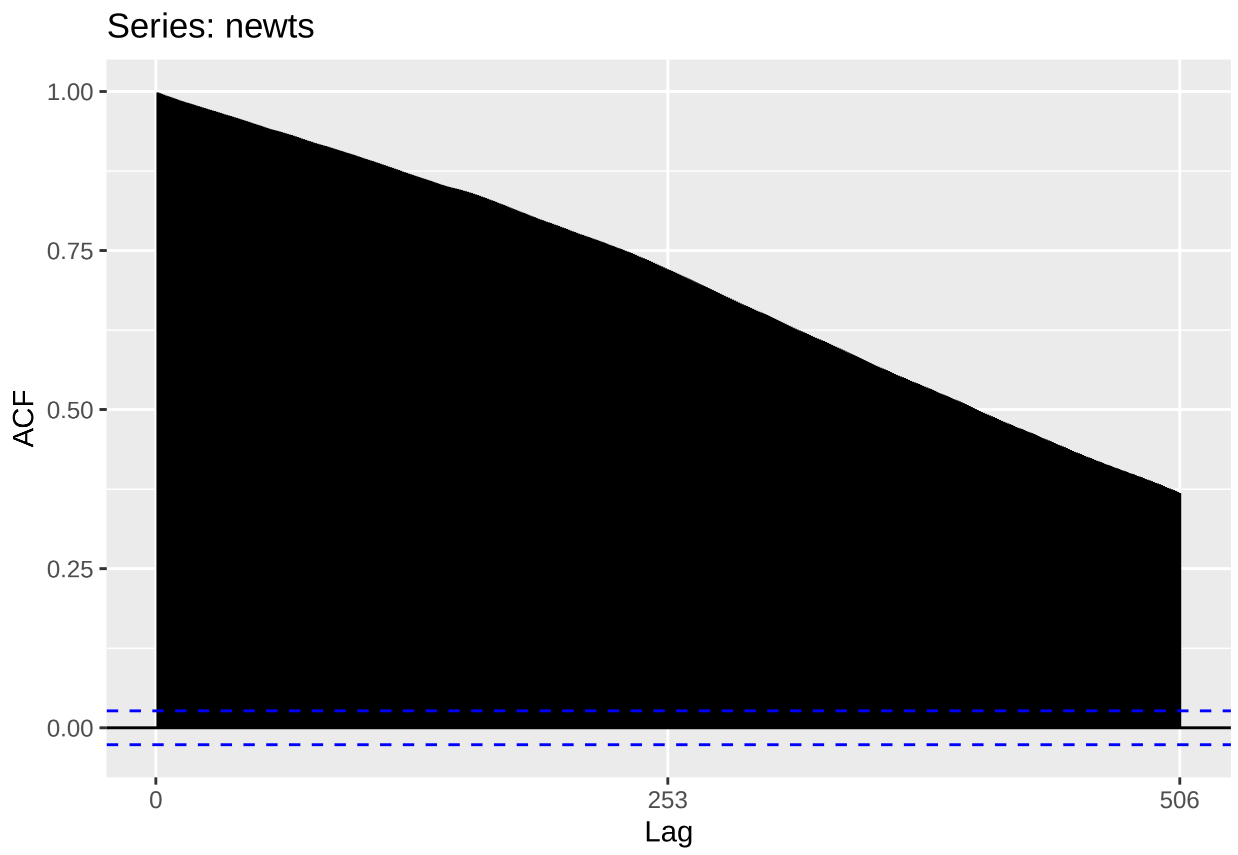
Figure 4: Bank of America daily adjusted closing stock price (1998-2019 without seasonality

Figure 4 shows the data after subtracting the seasonality component. After removing the seasonal component from the data, the data still appears to be very similar to before. The reason the data has not changed much is because the seasonal component contained numbers around zero. From the decomposition we can see that the seasonal coefficients are mostly between -0.5 and 0.5. These coefficients being small would explain why the data still looks similar after removing the seasonality.

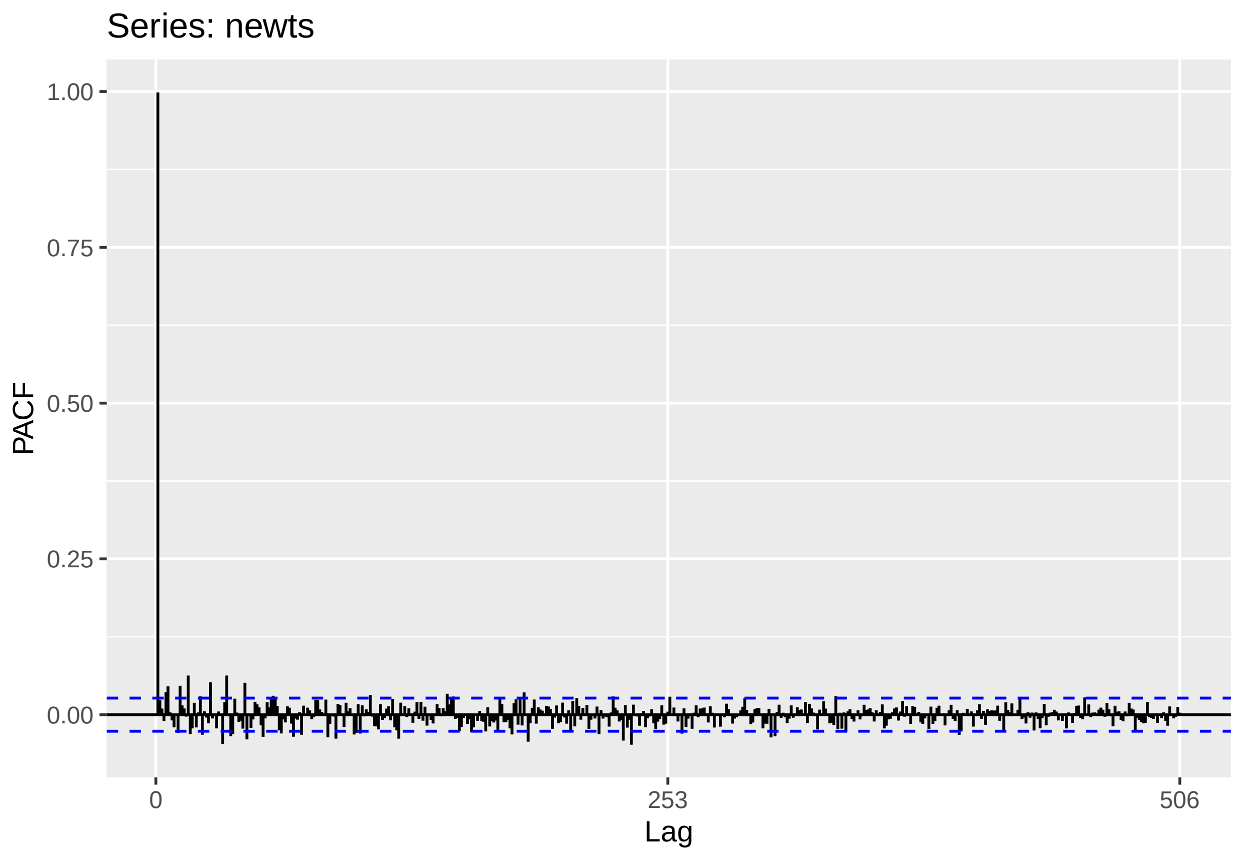
Autocorrelation

The autocorrelation (acf) and partial autocorrelation (pacf) plots for the data are shown below. These plots show the autocorrelations decreasing over time, but upon further inspection, the autocorrelations actually move in a very large sine-like wave. The pacf plot shows a significant partial autocorrelation only for the first lag with the partial autocorrelations much lower after the spike at the first lag.

```r
ggAcf(newts)
```



```r
ggPacf(newts)
```



```r
#acf2(newts)
```

ARIMA Model

Now it is time to build an ARIMA model. ARIMA stands for Autoregressive Integrated Moving Average model. These models are built by combining elements of both an Autoregressive model and a Moving Average model. In R, we need to use the `Arima()` function and tell it how many autoregressive parameters(`p`), number of differencing passes(`d`), and moving average parameters(`q`). Since I differenced once earlier to make the data stationary, we know that the differencing term should be one. The acf plot shows a sine-wave like pattern, and the pacf plot contains a spike only at lag one. These plots suggest that the moving average parameter should be zero. I will use Akaike information criterion (AIC) to determine the final model. The lower the AIC the better, so the model that has the lowest AIC should be the final model. After testing a few different numbers of autoregressive terms, I found that the model using five terms had the lowest AIC.

```r
mod <- Arima(tsobj, c(5, 1, 0))
summary(mod)
```

```
## Series: tsobj
## ARIMA(5,1,0)
##
## Coefficients:
##           ar1      ar2     ar3      ar4      ar5
##       -0.0302  -0.0122  0.0051  -0.0424  -0.0602
## s.e.   0.0135   0.0135  0.0135   0.0135   0.0136
```

```
##
## sigma^2 estimated as 0.222:  log likelihood=-3619.98
## AIC=7251.97   AICc=7251.98   BIC=7291.57
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE     MAPE       MASE
## Training set 0.00247472 0.4709599 0.2942962 -0.03514512 1.665688 0.05952353
##                       ACF1
## Training set -0.0004987368
```

The output summary shows that our final model will be:

$$X_t = -0.0302X_{(t-1)} - 0.0122X_{(t-2)} + 0.0051X_{(t-3)} - 0.0424X_{(t-4)} - 0.0602X_{(t-5)} + \varepsilon_{(t)}$$

The response variable is the stock price. In this model, the stock price is a function of the previous five days' stock price plus a random error. The MAE (mean absolute error) of the model is about 0.294. This error is relatively low, so the model seems to do a decent job explaining future stock price.

Forecasting

Figure 5 shows the model's forecasts for the next 500 days. For each day's prediction, the model will make use of the preceding five days of stock prices to forecast a predicted price. Using this model to forecast future stock prices shows that the model predicts prices to remain fairly constant over time. The model predicts the stock to barely change, but the confidence intervals grow wider over time. This growth occurs because we do not have data for years farther in the future, so we can only use increasingly lagged data to predict these values.

```r
mod %>%
  forecast(h = 500) %>%
  autoplot() + labs(y="Adjusted Closing Price", x="Date")
```

Since this dataset only contained data up to August 9, 2019, we can look at the real prices for the next week to see how the model actually fared. I found the stock data for the next week on Yahoo. The adjusted close variable I have been working with so far is the same as the close variable from this test data.

```r
test <- read.csv("BAC.csv")
head(test)
```

```
##         Date  Open  High   Low Close Adj.Close    Volume
## 1 2019-08-09 28.27 28.52 27.97 28.33  27.99168  53499600
## 2 2019-08-12 27.83 27.85 27.46 27.64  27.30992  52995900
## 3 2019-08-13 27.57 28.25 27.38 27.72  27.38896  69919200
## 4 2019-08-14 26.94 27.08 26.28 26.42  26.10449 106591700
## 5 2019-08-15 26.58 26.83 26.21 26.25  25.93652  70192000
## 6 2019-08-16 26.53 27.12 26.44 27.03  26.70721  70841700
```

```r
forecast(mod, h=5)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2019.486       28.38677 27.78288 28.99066 27.46320 29.31035
## 2019.490       28.39011 27.54889 29.23133 27.10358 29.67664
```

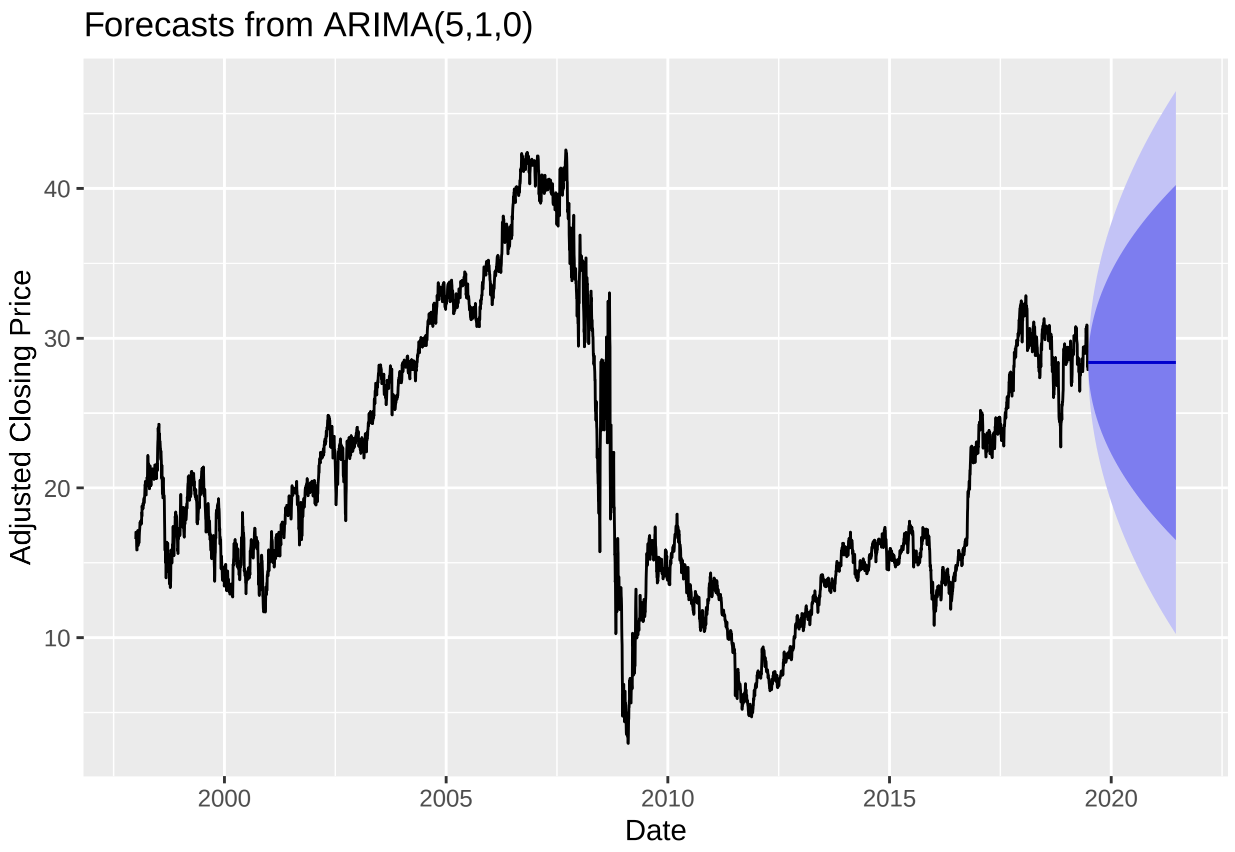


Figure 5: ARIMA forecasts for next 500 days

```
## 2019.494       28.40023 27.37912 29.42135 26.83857 29.96190
## 2019.498       28.37277 27.19730 29.54825 26.57504 30.17051
## 2019.502       28.37410 27.07351 29.67470 26.38502 30.36319
```

The model does not do too well here. The model's 95% confidence interval for the daily closing price only contained the actual price on three out of the five days I looked at here.

Intervention Analysis

One thing that stuck out in the initial plot of the data was the large drop in stock price during the Great Recession. This event looks like it may have had a permanent effect on the overall time series trend. To look at the effect this event is having on the data, we can perform intervention analysis. Since the Recession did not occur in a single day, I decided to just use the day that BAC experienced its biggest loss. This day would be 10/7/2008 where the adjusted closing price fell 7.71 points for about a 26% decrease in value.

To assess the effect of the intervention, the first thing we need to do is indicate where in our data the intervention occurs. Looking at the graph, the intervention seems to have a long-term effect on the stock's price. After the intervention, the stock appears to gradually return to the previous mean. This pattern suggests that we need a pulse at 10/7/2008.

```
neworder <- ordered
neworder$step <- ifelse(neworder$date == as.Date("2008-10-07"), 1, 0)
intmod <- arima(neworder$close_adjusted, order=c(5, 1, 0), xtransf=neworder$step, transfer=list(c(1,0))
summary(intmod)
```

```
##
## Call:
## arima(x = neworder$close_adjusted, order = c(5, 1, 0), xtransf = neworder$step,
##     transfer = list(c(1, 0)))
##
## Coefficients:
##           ar1      ar2     ar3      ar4      ar5  T1-AR1   T1-MA0
##       -0.0396  -0.0365  0.0049  -0.0108  -0.0086  0.9100  -7.2715
## s.e.   0.0136   0.0137  0.0136   0.0137   0.0140  0.0002   0.4746
##
## sigma^2 estimated as 0.2126:  log likelihood = -3503.86,  aic = 7021.73
##
## Training set error measures:
##                        ME      RMSE       MAE         MPE     MAPE      MASE
## Training set 0.002357429 0.4610053 0.2930681 -0.03476591 1.662108 0.9954629
##                       ACF1
## Training set -0.0005685369
```

The output summary shows that our final model will be:

$$X_t = m_t - 0.0396X_{(t-1)} - 0.0365X_{(t-2)} + 0.0049X_{(t-3)} - 0.0108X_{(t-4)} - 0.0086X_{(t-5)} + \varepsilon_{(t)}$$
$$m_t = 0.91m_{t-1} - 7.2715P_t$$

This new model can be interpreted similarly to the ARIMA model we found earlier. The coefficients have slightly changed, but the most important change here is the addition of the m_t, or transfer function. The transfer function will explain the effects of the intervention. The mean absolute error of this new model is about 0.293, which is almost the same as the model without intervention. The AIC of the model with intervention is slightly lower, though, so it might be a better model.

Estimating Effects of Intervention

The m_t function can be interpreted as a decrease of 7.2715 at the initial date of the intervention. After this initial pulse, the effects of the intervention on each subsequent day will be 91% of the previous day's effect. I tried plotting the effects of the intervention in the plot shown below.

```
neworder$pulse <- neworder$step*-7.2715
neworder$inteff <- NULL
for(i in 1:5436){
  neworder$inteff[i] <- neworder$pulse[i] + 0.91 * neworder$inteff[i-1]
  neworder$inteff[1] <- 0
}
```

```
## Warning: Unknown or uninitialised column: 'inteff'.

## Warning: Unknown or uninitialised column: 'inteff'.

## Warning: Unknown or uninitialised column: 'inteff'.
```

```
intts <- ts(neworder$inteff, start = c(1998), frequency = 253)
autoplot(intts) + labs(y="Effect of Intervention", x="Date")
```

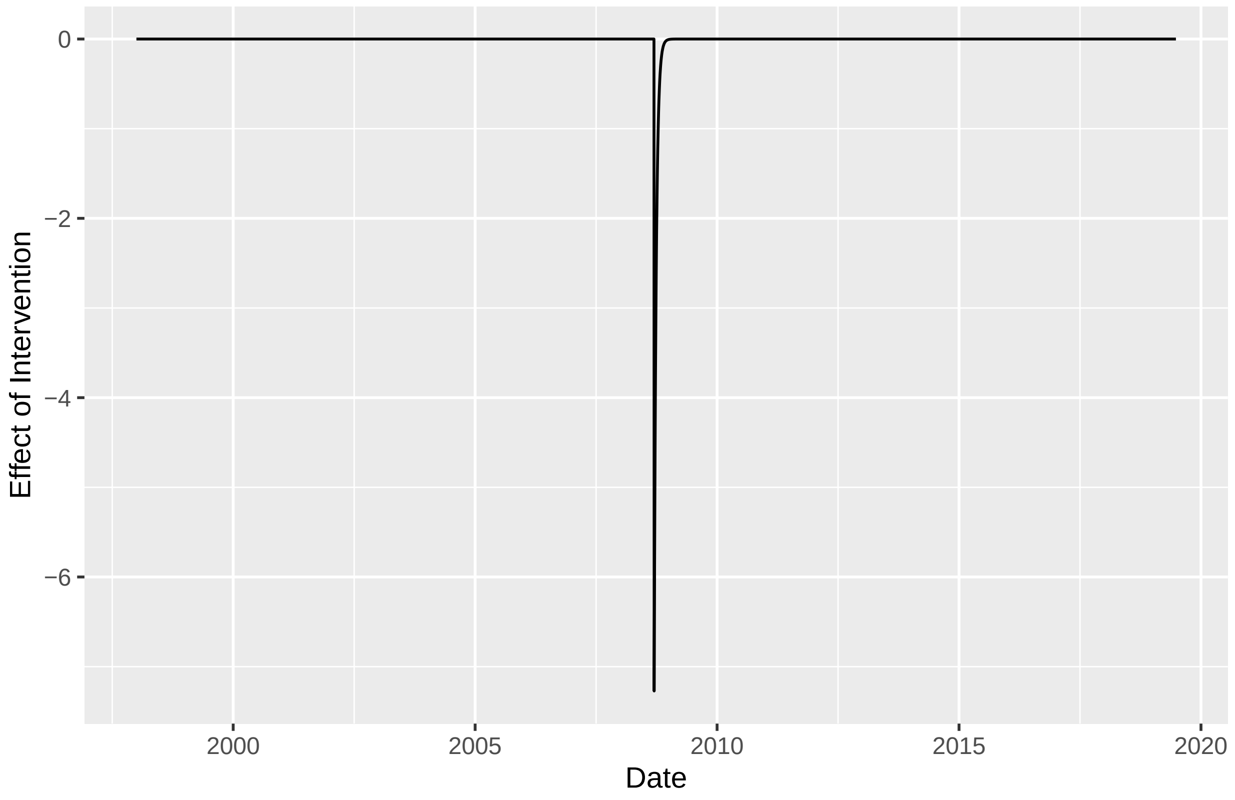


Figure 6: Effect of Great Recession on BAC Stock Price

As seen in Figure 6, the effects of the intervention are initially fairly large, but seem to go away rather quickly. Since the data were collected daily, even a 9% decrease in the effect takes a relatively short amount of time to go away. With 253 observations each year, the intervention effect basically has disappeared by the next year.

Findings and Conclusions

Generally, I found that Bank of America's stock price should stay relatively constant over time. The initial ARIMA model predicted that over the next 500 days, the stock would pretty much stay the same. When I examined the predictions for the week after the data ended, the model's 95% confidence interval only captured the actual stock price in three out of five days. As soon as the price changed by more than a few cents, the model was incorrect. A drop of only one dollar threw off the model's predictions. This problem seems to suggest that the model does a poor job explaining the volatility of the stock market. The issue here is that the model attempted to only explain stock prices through historical stock prices. It does not contain any way to account for any current events or firm actions that could affect the stock's valuation. To create a more accurate model, these types of variables would need to be accounted for.

The model I created with intervention analysis was very similar to my first ARIMA model. Upon plotting the actual intervention effects, the model seemed to severely underestimate the actual effects of the Great Recession. After only a few weeks, the effect of the intervention had become almost zero. I think the issue here is that intervention analysis requires an intervention at a single point in time. The actual economic recession spanned hundreds of observations in this dataset. Since the intervention effect decreases with each observation, the effect ended up being essentially zero before the recession had even ended. Unfortunately, I was not able to create forecasts with the intervention analysis. Upon doing some research, I found that there is currently no R package for forecasting with a transfer function approach. It would have been nice to visualize these forecasts, but, based off the plot of the intervention effects, the forecasts most likely would not have been very accurate.

Overall, these models seemed not to do a very good job of actually predicting future stock prices. This result is a little disappointing but not entirely unexpected. I think I would have been more surprised if my models were able to accurately predict future stock market prices. A model with the capability to correctly predict stock movements would be worth a fortune. Future extensions of this work would require some way to take into account a firm's actions and their effects on its stock price.

Works Cited

Alpha Vantage, Nasdaq, Yahoo Finance and Zacks (2019). US historical stock prices with earnings data. Retrieved from: https://www.kaggle.com/tsaustin/us-historical-stock-prices-with-earnings-data#stock_prices_latest.csv.

Cryer, Jonathan D., & Chan, Kung-Sik. (2008) Time Series Analysis With Applications in R, 2nd edition, Springer Science+Business Media: New York, NY. https://link-springer-com.ezproxy.amherst.edu/content/pdf/10.1007%2F978-0-387-75959-3.pdf. Accessed on January 10, 2020.

Hyndman, Rob J. "Forecasting using R," *Monsah University*, https://robjhyndman.com/eindhoven/3-2-Dynamic-Regression.pdf.

Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2. Accessed on December 10, 2019.

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2019). *forecast: Forecasting functions for time series and linear models*. R package version 8.9, <URL: http://pkg.robjhyndman.com/forecast>.

Rob Hyndman, Alan Lee, Earo Wang and Shanika Wickramasuriya (2018). hts: Hierarchical and Grouped Time Series. R package version 5.1.5. https://CRAN.R-project.org/package=hts

"9.2 Intervention Analysis: STAT 510," *Penn State Eberly College of Science*, https://online.stat.psu.edu/stat510/lesson/9/9.2.

Wathen, Jordan. “Bank of America Stock Split History,” *The Motley Fool* (February 27, 2018), https://www.fool.com/investing/2016/10/25/bank-of-america-stock-split-history.aspx.

Yahoo Finance. (2019). Bank of America Corporation (2019). Retrieved from https://finance.yahoo.com/quote/BAC/history?period1=1565308800&period2=1566000000&interval=1d&filter=history&frequency=1d.