Forecasting Apple Stock Price through Time-Series Analysis

Debbie Tan

Introduction to Data Science

Instructor: Daniel D. Gutierrez

UCLA Extension – Spring 2021

1 Introduction

The US stock market has seen a huge influx of retail investors and traders over the past year, especially over the past couple months following the Gamestop/hedge fund saga. As one of these new investors, I was interested in whether I could forecast stock prices for my own investment purposes.

Machine learning is the core basis of “algo trading” as people call it these days. Entire financial institutions are built around these predictions that incorporate transaction data and technical analysis indicators to inform the optimal prices and times to buy and sell securities. One example is the brokerage app Robinhood which is extremely popular amongst retail investors—this company collects all the transaction from its users and sells it to Citadel and other market makers. This information is incredibly insightful as these money markets can harness this data to more properly analyze momentum & sentiment, as well as better anticipate investors’ next trades.

I initially attempted to include technical indicators into my analysis, but at my current skillset level I had to resort to using existing time-series analysis algorithms for prediction with the Apple stock price.

Data Set

I imported the Apple stock price data from 2018 onward using an R package called quantmod, which by default scrapes this data from Yahoo Finance.

I used summary(), head(), and str() to take an initial look at the data set.

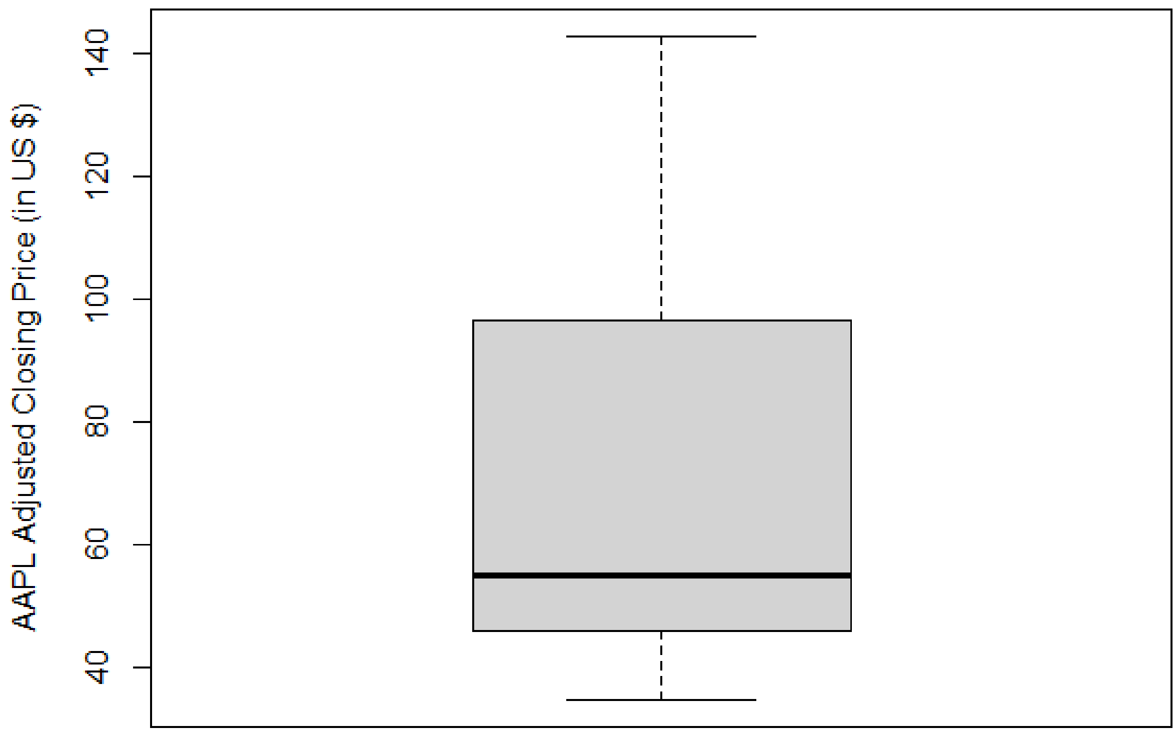
```
> summary(AAPL)
     Index              AAPL.Open         AAPL.High          AAPL.Low         AAPL.Close       AAPL.Volume          AAPL.Adjusted
 Min.   :2018-01-02   Min.   : 35.99   Min.   : 36.43   Min.   : 35.50   Min.   : 35.55   Min.   : 45448000   Min.   : 34.61
 1st Qu.:2018-11-06   1st Qu.: 47.10   1st Qu.: 47.50   1st Qu.: 46.71   1st Qu.: 47.15   1st Qu.: 90104400   1st Qu.: 45.78
 Median :2019-09-17   Median : 56.25   Median : 56.83   Median : 55.62   Median : 56.10   Median :114731400   Median : 54.94
 Mean   :2019-09-16   Mean   : 72.63   Mean   : 73.49   Mean   : 71.78   Mean   : 72.66   Mean   :131078821   Mean   : 71.73
 3rd Qu.:2020-07-24   3rd Qu.: 96.99   3rd Qu.: 98.11   3rd Qu.: 95.94   3rd Qu.: 97.01   3rd Qu.:154465725   3rd Qu.: 96.37
 Max.   :2021-06-02   Max.   :143.60   Max.   :145.09   Max.   :141.37   Max.   :143.16   Max.   :426510000   Max.   :142.70
```

```
> head(AAPL)
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2018-01-02   42.5400   43.0750  42.3150    43.0650   102223600      41.31007
2018-01-03   43.1325   43.6375  42.9900    43.0575   118071600      41.30288
2018-01-04   43.1350   43.3675  43.0200    43.2575    89738400      41.49474
2018-01-05   43.3600   43.8425  43.2625    43.7500    94640000      41.96716
2018-01-08   43.5875   43.9025  43.4825    43.5875    82271200      41.81128
2018-01-09   43.6375   43.7650  43.3525    43.5825    86336000      41.80650
> str(AAPL)
An ‘xts’ object on 2018-01-02/2021-06-02 containing:
  Data: num [1:860, 1:6] 42.5 43.1 43.1 43.4 43.6 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:6] "AAPL.Open" "AAPL.High" "AAPL.Low" "AAPL.Close" ...
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
List of 2
 $ src    : chr "yahoo"
 $ updated: POSIXct[1:1], format: "2021-06-03 21:04:14"
```

From here we can see that there are no NAs, which is great! We don’t have to account for those. We also see that the data set is being pulled in as an XTS object, so we do not need to coerce dates from character strings to date class data.

2 Exploring the Data (EDA)

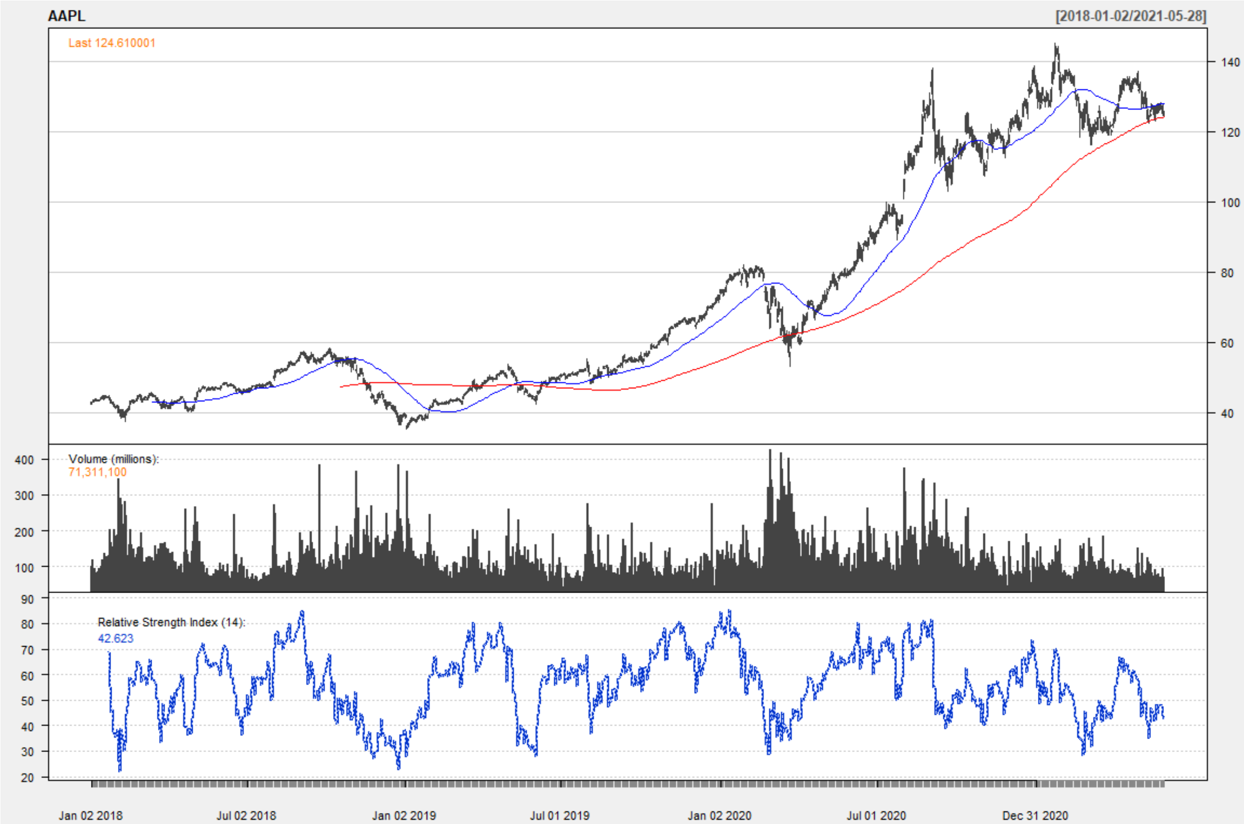
Box Plot
Let's do a box plot to get visualize the distribution of data. We'll plot it for adjusted closing price, which accounts for stock splits, and what we'll ultimately want to predict.



The boxplot shows that there are no extreme outliers in the data set, so we don't necessarily need to omit any numbers at this point. It's important to note that the data does seem skewed to the lower $40-100 range. We're seeing a long upper whisker, which is most likely attributed to the meteoric rise in price during 2020! We may need to exclude 2020 as an outlier/anomaly if the model does not work out.

Let's take a look at Apple's closing price plotted over time with technical indicator overlays (50/200 day SMAs and RSI). I used the chartSeries() function from the quantmod package for this.

Time Series Graph with Technical Indicators



During an uptrend, RSI (relative strength index) tends to stay above 30 and should frequently hit 70. During a downtrend, it's rare to see RSI exceed 70, and should frequently hit 30 or below. Since the beginning of 2021, we've seen large upswings and downswings in RSI—we see it rise above 70 and go below 30 about once each during 2021, and current RSI is around 43. This may suggest that the price is stabilizing at the moment.

Simple moving average (SMA) is another commonly used technical analysis tool that helps smooth out price data by taking the average over a specific period of time. A frequently used bullish signal is when the 50-day SMA (blue line in the top portion of the chart above) crosses above the 200-day SMA (red line)—this is also known as a "golden cross" moment. On the other hand, a bearish signal is when the 50-day SMA crosses below the 200-day SMA, also known as a "death cross." We see a golden cross around June 2019, the start of a 6 month long bullish period, followed by the March 2020 COVID crash and an extremely bullish period. We also see a death cross around late 2018, but that was a brief bearish period. Most recently, we're seeing the SMAs converging but not completely touching. This may be another indicator that we're finding a new floor stock price.

2 Data Transformation and Pre-Processing

Testing for Stationarity

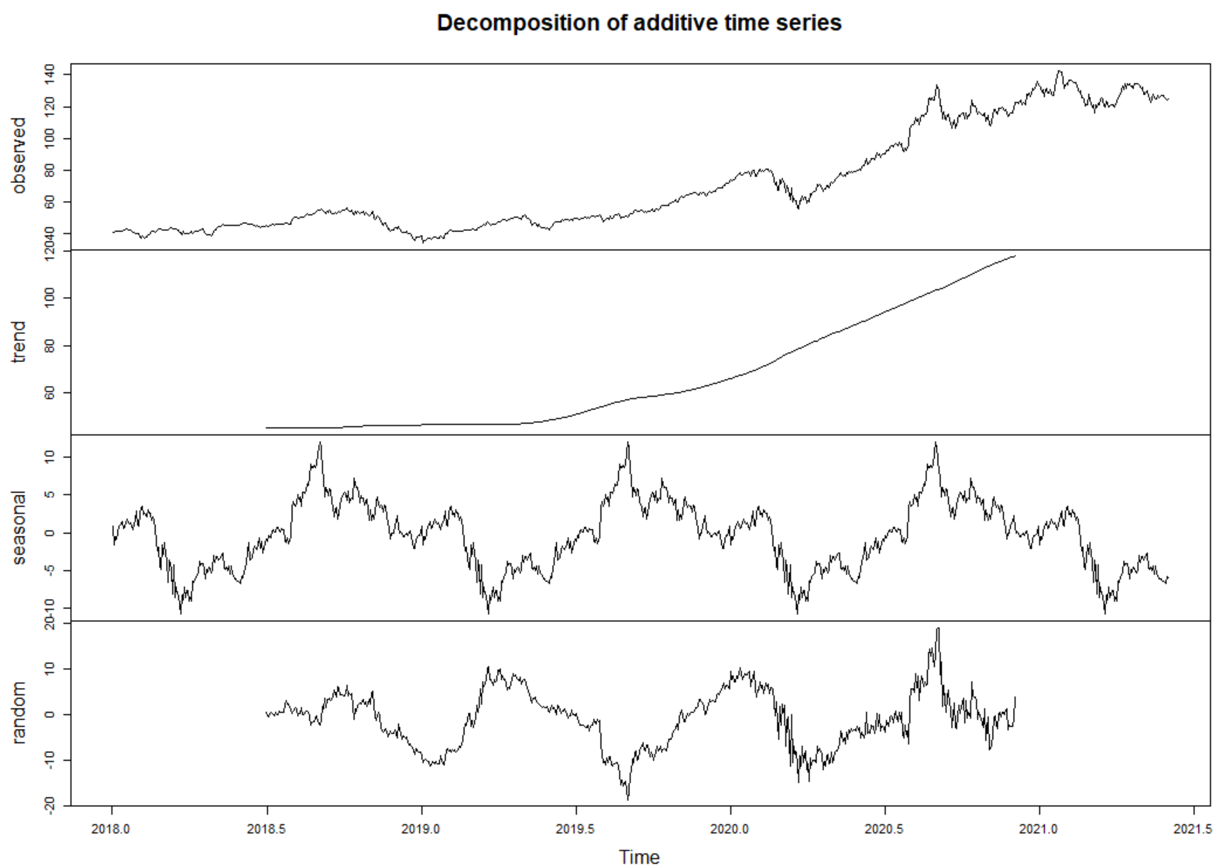
To run time series models with the ARIMA model (Auto Regressive Integrated Moving Average), we must adjust for non-stationarity. In other words, we must have a flat-looking series that has constant variance over time and no seasonality to run certain kinds of ARIMA models. If the initial data set shows non-stationarity we adjust for that—depending on the kinds of adjustments we make, we will select a certain ARIMA model to use for forecasting.

First, I used the Augmented Dickey-Fuller Test to check for stationarity.

```
> adf.test(AAPL$AAPL.Adjusted)

        Augmented Dickey-Fuller Test

data:  AAPL$AAPL.Adjusted
Dickey-Fuller = -2.0226, Lag order = 9, p-value = 0.5687
alternative hypothesis: stationary
```

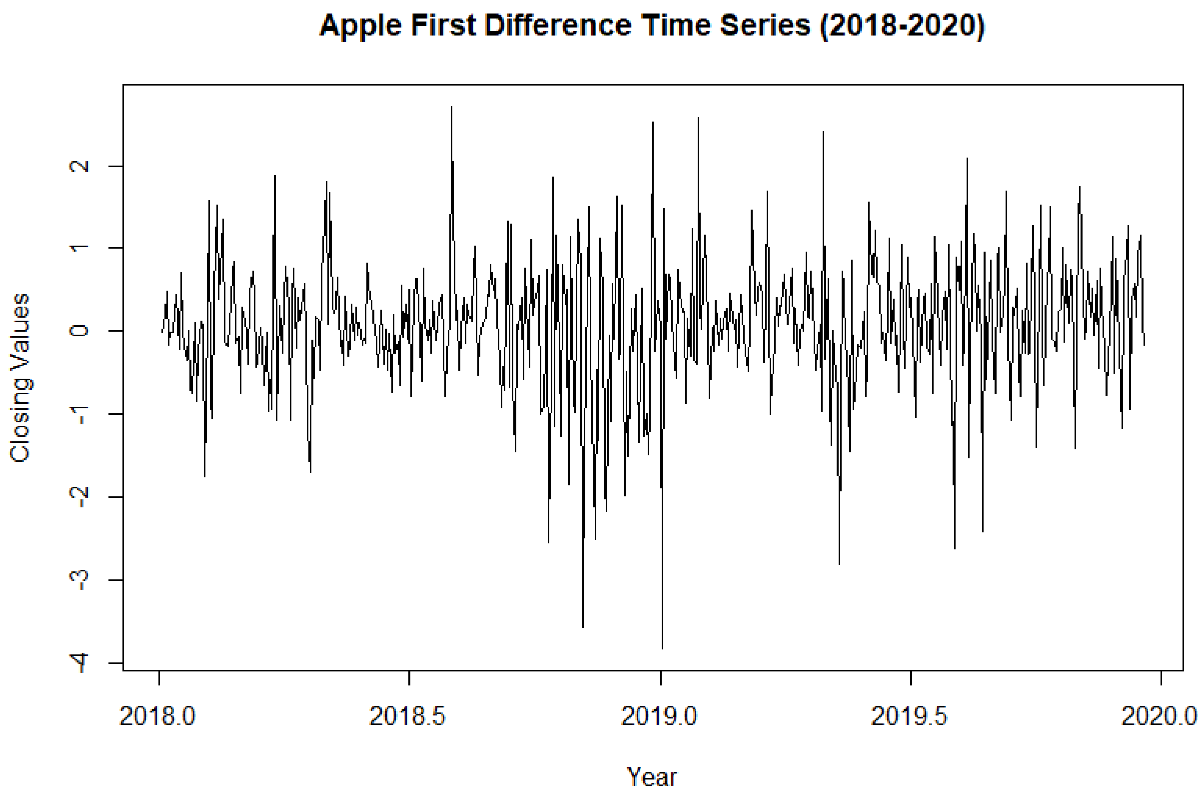
Since the p-value is well over 0.05, we fail to reject the alternative hypothesis of stationarity (in other words, we “accept” the null hypothesis that the data set is showing non-stationarity).

Let’s decompose the data set to see if there are any trends we can discern.



From the decomposed time series of Apple stock price, we can see a repeated pattern in the seasonal section, which means there may be a component of seasonality that we may need to adjust for before proceeding with the time series model.

To do this, we'll take the difference in the training set's daily closing values to account for the non-stationarity, then plot it for visualization.



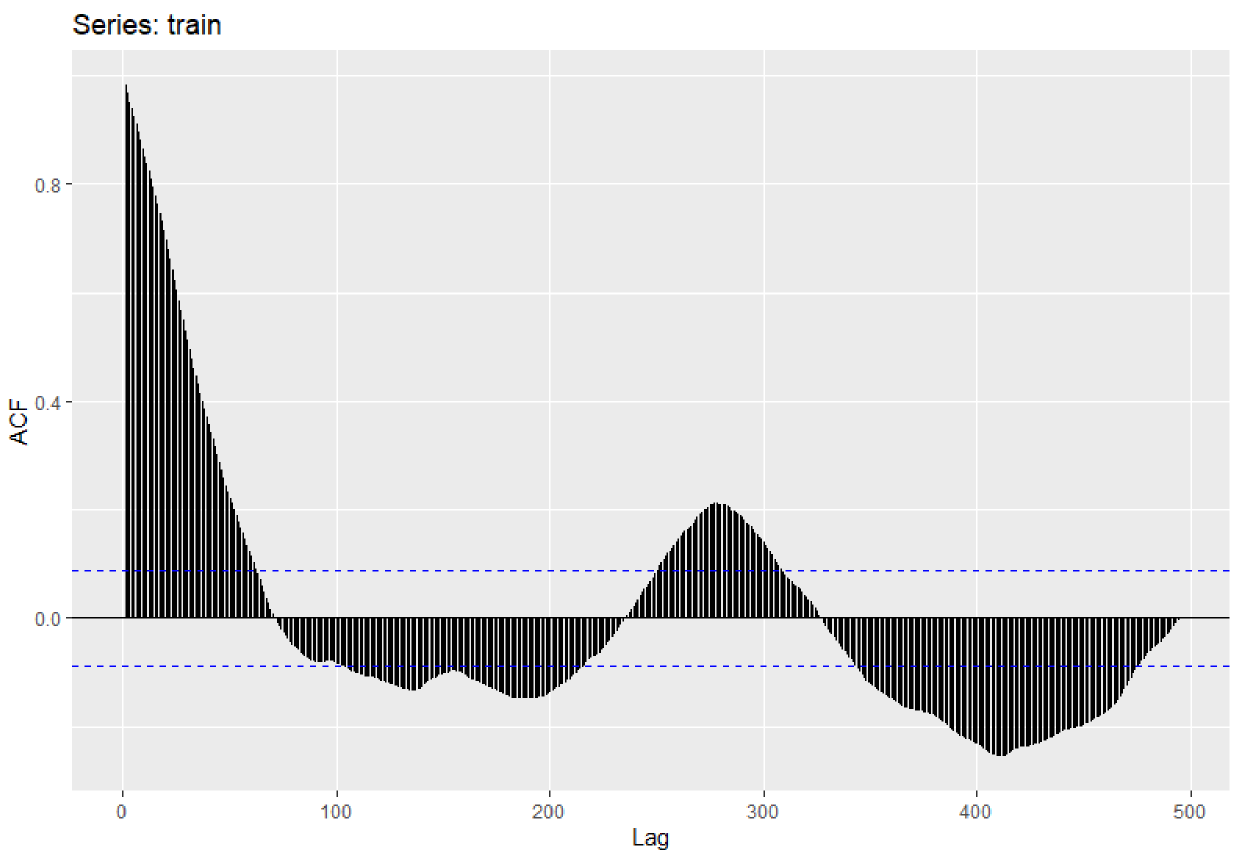
Overall, the mean and variance are more or less constant, and there look to be spikes (bigger differentials) every quarter or so.

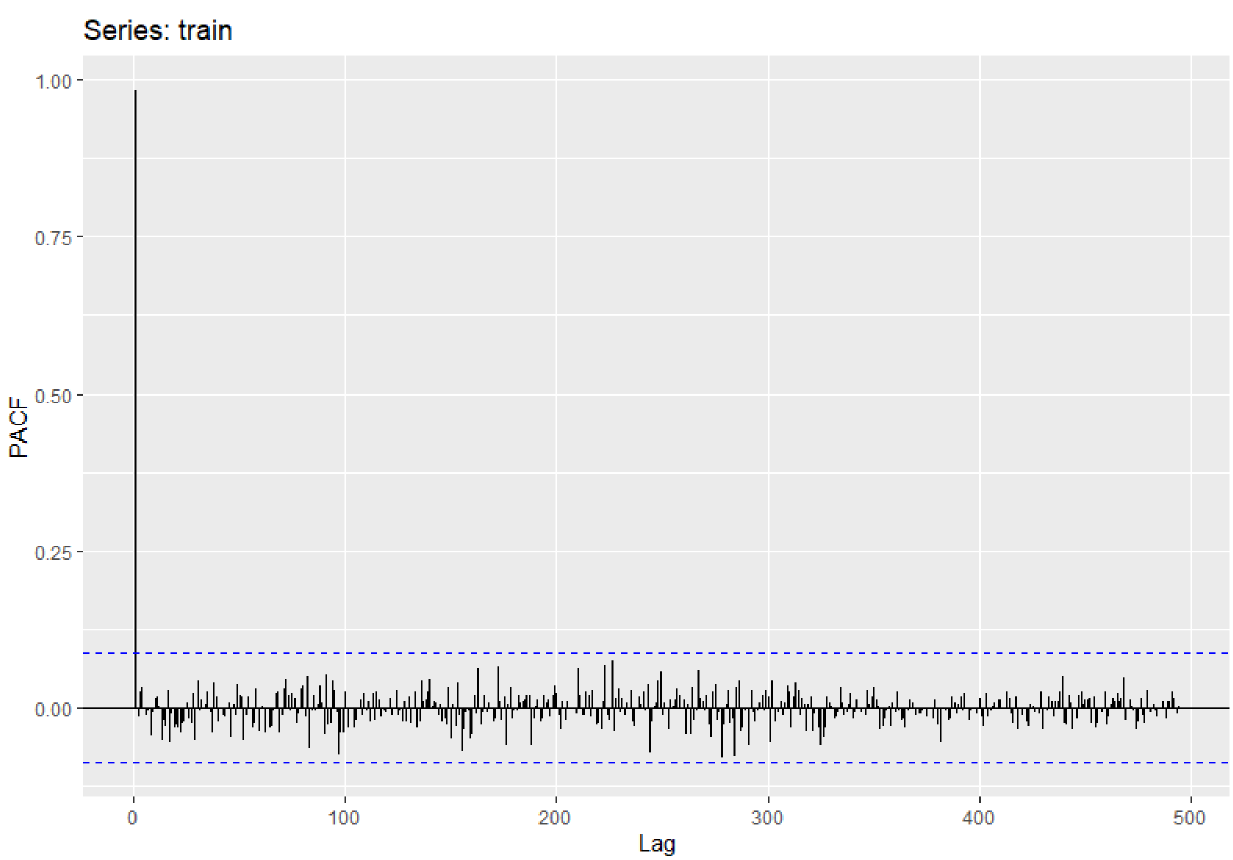
Now we'll run the ADF test again to test for stationarity with the first difference time series.

```
> adf.test(tsDiff) #Run the ADF test again to test for stationary data

        Augmented Dickey-Fuller Test

data:  tsDiff
Dickey-Fuller = -7.1262, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

p-value is well under 0.05, so we reject the null hypothesis and accept the alternative hypothesis that the first difference time series data set is stationary.

We'll run further visualizations to test for stationarity.


Series: train
0.8
0.4
0.0
ACF
0
100
200
300
400
500
Lag


Series: train
1.00
0.75
0.50
0.25
0.00
PACF
0
100
200
300
400
500
Lag

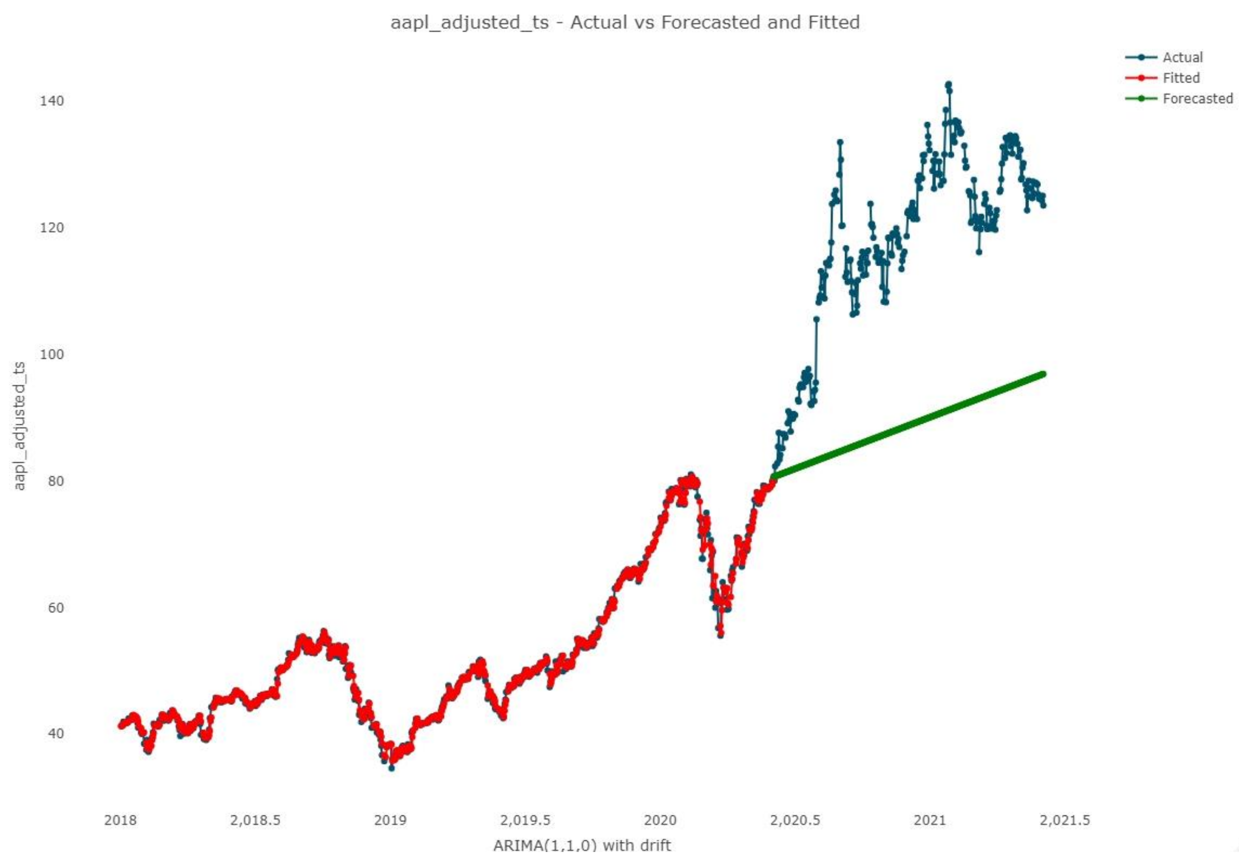
The ACF plot shows a decreasing trend while the PACF cuts immediately after 1 lag. With one order of differencing AR(1) is suggested, leading to an ARIMA(1,1,0) type of model.

3 Machine Learning Algorithm to Predict Closing Price

Building the Model and Viewing its Output

We will use supervised machine learning techniques to predict the adjusted closing price of the Apple stock. This involves training an algorithm with a portion of the data set we've been using (training set), then comparing the predicted values from the algorithm's output to the "test set" which contains the actual Apple stock closing prices over time. Based off how well the model can predict Y (adjusted closing price) from a given X (time/date), we will either keep the model as is or adjust the inputs and/or chosen algorithms to reduce error and get to a better prediction.

Using the auto.arima() function from the forecast package, we will select the best-fit ARIMA model for our data. Then using the test_forecast() function, also from the forecast package, we will plot the actual vs. fitted vs. forecasted data sets.



The function auto-selected the ARIMA(1,1,0) model, which corroborates our findings with the ADF tests and ACF/PACF plots. However, we can see that the forecasted values are well under the actual closing price values, especially during the latter half of 2020.

Checking the Accuracy of the Model

Using the accuracy() function within the forecast package, we see that there is a large discrepancy between the training and test sets' RMSE (standard deviation of residuals, AKA prediction error). The test set's RMSE is much higher than the training set's, indicating that we have overfit the data and should adjust the model.

```
> round(accuracy(forecast, test))
             ME RMSE MAE MPE MAPE MASE ACF1 Theil's U
Training set  0    1   1   0    1    0    0        NA
Test set     30   32  30  25   25    2    1        12
```

Future Considerations

Based off the actuals vs. fitted vs. forecasted plot we saw above, this makes sense. I didn't have time to adjust the model, but there are definitely a few things I would change for the next iteration. First, I would definitely exclude 2020 data from the training set. That year was such an anomaly due to COVID scares, resulting in the huge March 2020 crash—that coupled with a political administration that basically let the stock market run amuck during the rest of the year led to an unprecedented, and incredible, rise in stock price.

Another thing I would change for future iterations is add in technical indicators as feature variables. This may help train the model to more accurately predict and represent trends.