Sergio Gutierrez

Forecasting Stock Prices with ARIMA and Prophet Time Series Analyses

Problem Statement:

The US stock market offers many lucrative investment opportunities for both institutional and individual investors, barring any disastrous or unforeseeable changes with the economy or any specific publicly traded company/companies. The stocks offering the most returns to investors usually are from companies who have and continue to trend positively, both in financial performance and future outlook. Being able to forecast a stock's future trend and quantify its future value empowers investors to make more informed decisions whether to sell a stock now to maximize returns or buy more of it for even greater returns.

The goal of my project will be to use stock market data to determine the future trend and closing price of the Apple Inc. stock (denoted as AAPL). Through the Arima and Prophet time series models, I will be able to forecast its closing price for one year (from October 2018 to September 2019) and then compare the results between actuals and modelled forecasts. Additionally, I will be running the analysis on R as well to compare the forecast results and accuracies between the two programming languages.

Dataset Description:

I will be retrieving data from Yahoo! Finance for this project. Using the yfinance Python package and tidyquant R package, I will be able to download market data from Yahoo! Finance for Apple's stock. The data downloaded provides dates, symbols, volume, close price, adjusted close price, open price, high price and low price. With regards to the period and scope used for this project, I will be downloading ten years worth of Apple's stock market data, from October 2008 to September 2019. Note, there are about 250 trading days in a year as the market is not open on weekends (excluding Fridays).

Data Cleaning and Wrangling:

The following were steps to clean and prepare the data for time series analysis and modeling on Python:

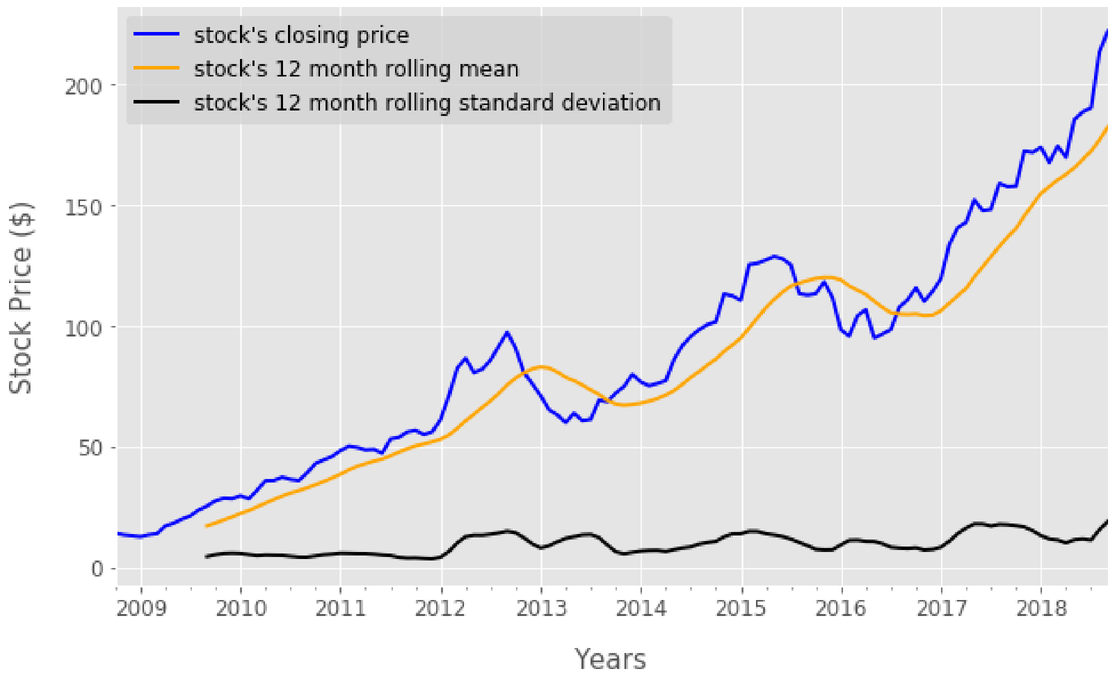
1. Upon downloading Yahoo! Finance data for Apple and a couple of other tech stocks with the yfinance Python package, the resulting data frame needed to be resampled from daily absolutes to monthly averages to reduce the volatility from analyzing and forecasting daily prices changes.
2. First, the data frame is inspected for any missing data points for any columns and fortunately all rows for each column have the same number of data entries.
3. The columns's data types are then inspected to ensure they are appropriately classified as numeric types, and fortunately they were classified as floats.
4. Since the package downloaded two column levels with the first being the stock market metrics and the second being the ticker symbols nested right under, I unstacked the ticker symbols to be their own column to make it easier to index specific metric column and symbol rows.
5. To make it even easier to index or reference columns, I made all column names lowercase.
6. I moved the resampled dates column to the row index because it will be easier to use and plug-in such any series and its accompanying datetime index in time series models.
7. The cleaned data frame was then partition into train and test data frames, with the train data covering the previous 9 years (from October 2008 to September 2018) and test data covering the latest year (from October 2018 to September 2019).
8. Lastly, the train and test data frames will be sliced to provide only Apple data rows.

Exploratory Data Analysis:

Apple's Closing Price Performance Over the Years

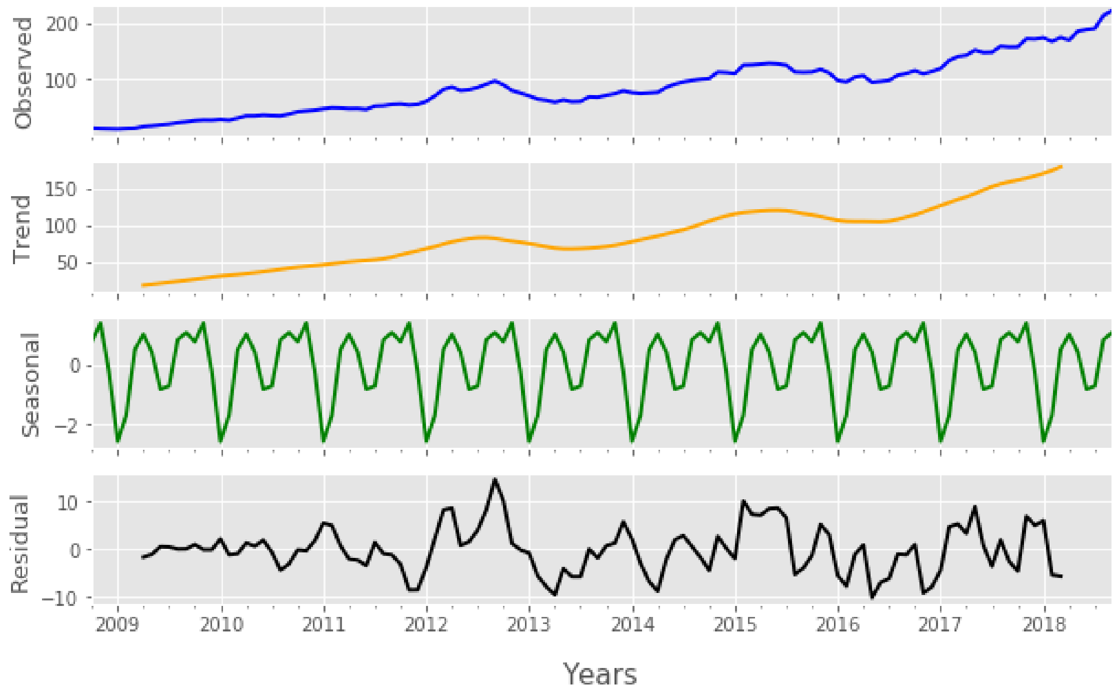
Based on its monthly performance over the previous nine years, Apple stock's closing price has been growing consistently despite two troughs in 2013 and 2016. With regards to the June 2013 stock price drop, market speculation of slowing smartphone sales growth from previous quarterly earnings reports sent the stock price downward. Similarly, in June 2016, the stock price dropped again due to soft iPhone sales especially when compared to those in 2015, which sent market speculation to lower the valuation.

Nevertheless, our 12 month rolling mean plotted below smooths out such large and any small market fluctuations to illustrate the long-term growth trend of Apple's stock price. On the other hand, 12 month rolling standard deviation below serves to illustrate the market volatility, or the degree of variation the stock price fluctuates from the mean. For the purposes of this project, we will not dive further in market volatility.



Seasonal Decomposition of Apple's Closing Price

Apple's stock price performance can be broken down and explained by three main time series components-- trend, seasonality and noise (as illustrated below).



To begin examining the breakdown of the time series decomposition above, the observed plot above is the same as the stock close price plotted before. As for trend, it is the overall increase or decrease seen over time. Similarly to the observed plot, the trend plot above is the same as the 12 month rolling mean plotted before. Seasonality is the repeating patterns or cycles within trend seen over time. Lastly, noise is the unaccounted, unexpected and remaining variability that cannot be attributed to either trend or seasonality.

As it relates to Apple's stock performance, the trend has been increasing, seasonality has shown fall and winter seasonal spikes during the months of Q3 and Q4 (coinciding with Apple fall keynotes and subsequent iPhone launches), and noise is not readily visible but we have explained the biggest noises causing change in price in both June 2013 and 2016.

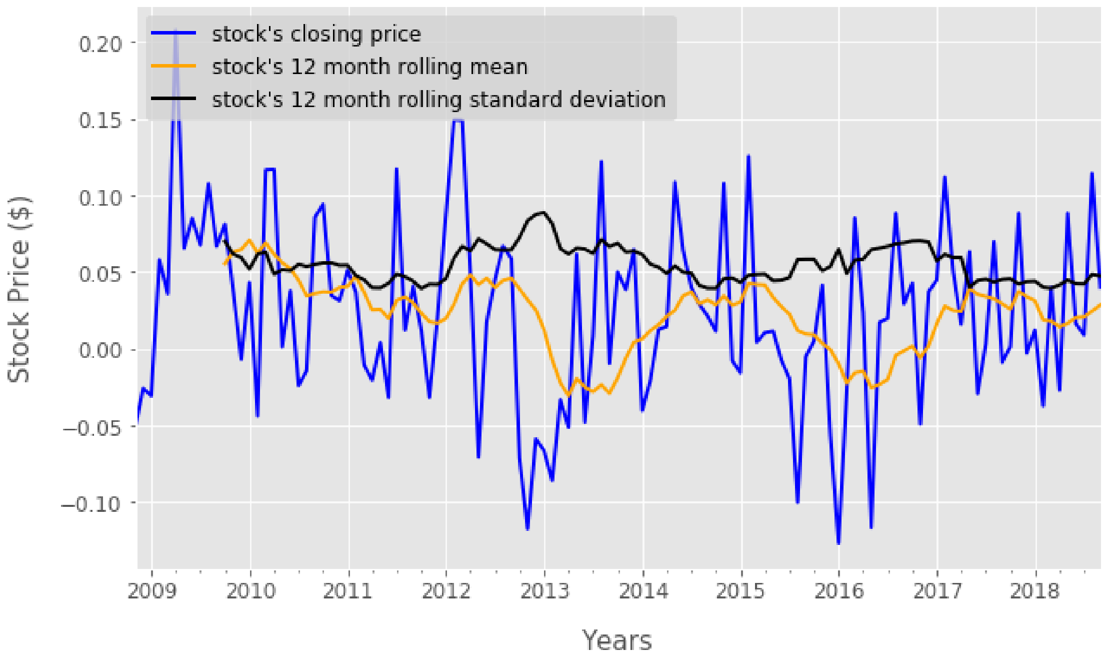
Apple Stock's Non-Stationary Conversion to Stationary

As we have seen in the previous two sets of graphs, Apple's stock price has been relatively positive and linear, where price has increased as time has increased. Despite the fluctuations in price over the months and years, the stock price has a positive relationship with time and follows a consistent trend and seasonality. These three components interfere with doing time series analyses because the primary and most important assumption to using time series techniques is the observations provided are stationary, suggesting they not dependent on time and devoid of trend and seasonality. Statistically, stationarity also means the series being used is normally distributed, and the mean and variance are constant over time.

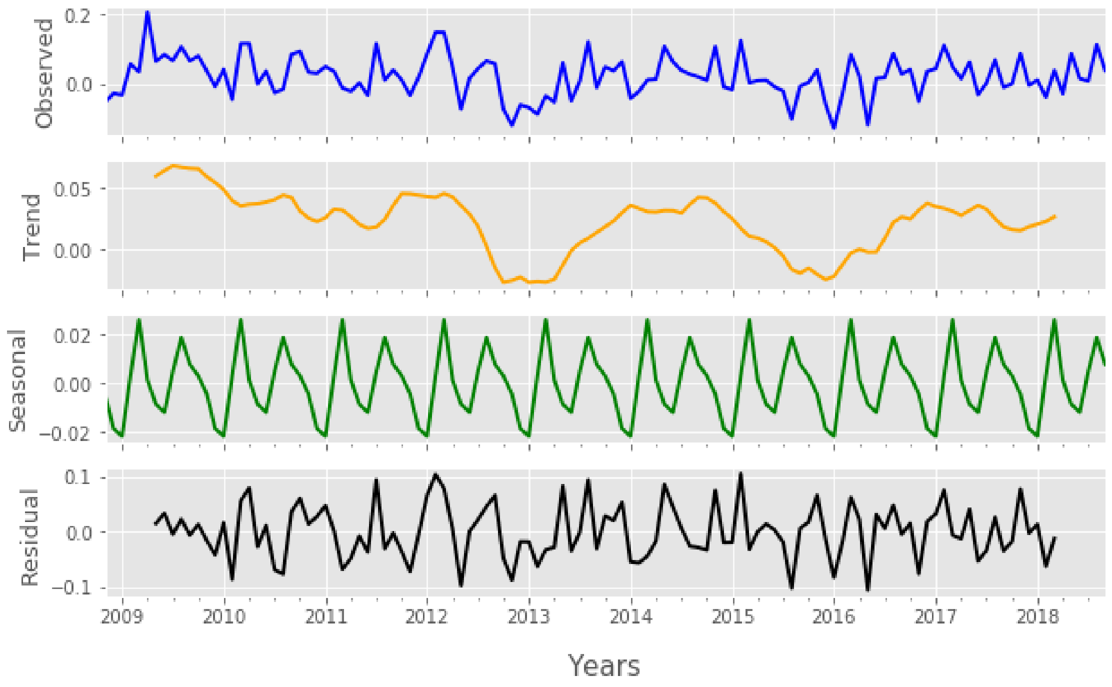
In addition to visually demonstrating Apple stock not being stationary, we can also statistically test whether or not the data is stationary using the Augmented Dicky-Fuller (adfuller) test. The test's null hypothesis states a time series has a unit root and thus is non-stationary. It's alternative hypothesis states the time series does not have a unit and thus is stationary. Using the adfuller test on the Apple stock time series, the output below confirms that our p-value (of 0.99) is nowhere near close to or below the significance level of 0.05 and thus we fail to reject the null hypothesis (calculated below).

Results of Dickey-Fuller Test:	
Test Statistic	1.685822
p-value	0.998094
#Lags Used	0.000000
# of Observations Used	119.000000
Critical Value (1%)	-3.486535
Critical Value (5%)	-2.886151
Critical Value (10%)	-2.579896

However, the data can be transformed to erase this dependence on time and eliminate trend and seasonality. First, we implement a logarithmic transformation on the closing price to turn the absolute variations of the time series to be one of relative variations, ensuring that any two equivalent price changes are represented by the same vertical distance on a logarithmic scale. Next, to eliminate trend and seasonality we implement a differencing transformation in which we subtract the previous observation from current observation. In effect, this makes each monthly observation uncorrelated with the previous ones and thus eliminating any relationship the closing price has with time, trend and seasons (pictured below).



The resulting transformation eliminated non-stationarity for time series and trend, and reduced it for seasonality (as shown below). For instance, we can see the observations have no discernable relationship between price and time. Additionally, we can see that the trend component has turned from positive linear to sinuous curve without any continuous increases or decreases over time. Lastly, we can see the seasonality duration and monthly peaks have been reduced.



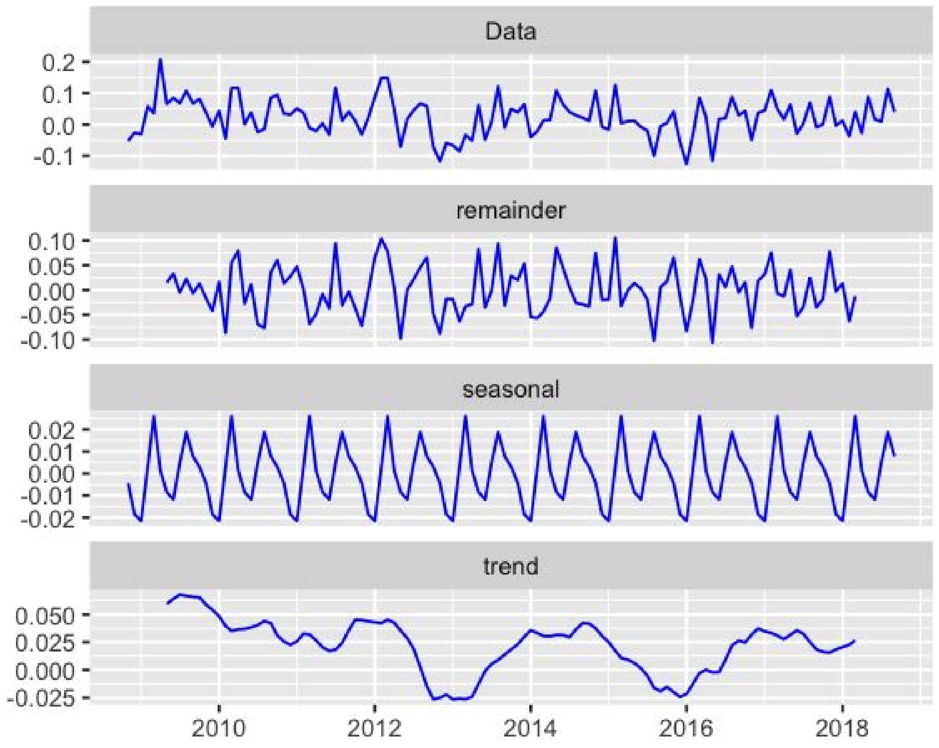
To ensure we have statistically met the requirement of stationarity, we will run the adfuller test once more on the transformed data. As shown below, our p-value is extremely small and well below our significance level (0.05), allowing us to reject the null hypothesis and accept the alternative that our series is stationary. With this crucial assumption met and our data preparation set, we can move forward with building our ARIMA and Prophet time series models, but before we do let’s compare our results with R programming.

Results of Dickey-Fuller Test:	
Test Statistic	-8.235947e+00
p-value	5.889858e-13
#Lags Used	0.000000e+00
# of Observations Used	1.180000e+02
Critical Value (1%)	-3.487022e+00
Critical Value (5%)	-2.886363e+00
Critical Value (10%)	-2.580009e+00

How Do the Seasonal Decomposition and Adfuller Test Compare on R?

Given that the raw data and data transformation will appear the same regardless of the language used, I will be skipping to the time series applications and testing results.

As we might have expected, the seasonal decomposition of Apple's closing price on R is very similar to that of Python, if not identical. The arrangement of the components is different and with slightly different naming conventions for the components (i.e., "data" instead of "observed" and "remainder" instead of "residual") but everything is essentially the same in both behavior and magnitude for all three components.



Moving on to the adfuller test, we again receive a p-value smaller than our significance level (0.05), allowing us to reject the null hypothesis and accept the alternative that our data is now stationary.

Augmented Dickey-Fuller Test	
data:	stock_diff
Dickey-Fuller =	-8.3389,
Lag order =	0,
p-value =	0.01
alternative hypothesis:	stationary

All in all, I did not expect to see any major differences in our initial results and testing when using Python and R. However, I do anticipate seeing more differences when forecasting and measuring the accuracy of our forecasts because the methodologies used and statistical packages defined should be different between both languages. Nevertheless, the results from both will be both enriching and helpful for forecasting stock prices for both institutional and individual investors.