EXPLORING TIME SERIES ANALYSIS IN R, AND INTEGRATION WITH TABLEAU

Contents

CHAPTER 1: INTRODUCTION

Time is considered by every business in one form or another and is one of the most important factors when considering one's success: what will our sales be next month, or next year? Annual crop production? Website traffic over time and much more! Technology has advanced so quickly in recent years that now businesses have the power to 'predict' the future.

Time Series Analysis is one such method applied by businesses in order to understand the past to then try and predict the future, after all "time is money"! Business managers at all levels understand this phrase and inevitably make important decisions under uncertainty, i.e. they forecast. Time Series Analysis involves the statistical analysis of a time series of data which can be described as a sequence or collection of data points that is measured over time at specific intervals. Time series analysis is the practise of using historic data to try and identify trends or patterns in that data to forecast future values in the series.

This project is concerned specifically with time series forecasting and the understanding and application of various forecasting models such as the below:

- Autoregressive (AR)
- Moving average (MA)
- Autoregressive moving average (ARMA)
- Autoregressive Integrated Moving Average (ARIMA)
- Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

Requirements Specification

Project Goal: How accurate and reliable can modern time series analysis models be in predicting the future, and in particular stock market prices?

To answer this question I researched, developed and applied multiple time series models on a number of stock market indexes, and measured their accuracy (predicted vs actual). I implemented these models using the free open source statistical programming software R. I also integrated R with Tableau to utilise Tableau's excellent visual capabilities and experiment with Tableau's built in exponential smoothing forecast function.

When compared to a regular regression problem time series modelling can be an ambitious task. The famous quote says it all "prediction is very difficult, especially about the future". Unlike "ordinary" data time data can contain trends and have a seasonal element plus the order of the data in the series is one of the most important factors to consider.

Finally it is worth noting that time series forecasting can be considered a form of technical analysis, meaning that it is assumed that all the information required is wholly contained within the stock price itself.

CHAPTER 2 – BACKGROUND & LITERATURE REVIEW

Understanding Time Series

Time series have 3 main features:

- **Autocorrelation or serial correlation/dependence**: Time series related observations that are close together in time are usually correlated. Daily returns for example:

r_1	r_2	r_3	r_4	⋯	r_n
daily return	daily return	daily return	daily return		daily return

- **Trends:** a trend is the general direction of a market or a price of a given asset (stock).
- **Seasonality:** A time series would have seasonality when patterns it may have follow a calendar effect (monthly, quarterly, yearly) repeatedly, e.g. temperatures rise and fall in a similar pattern every year from season to season.

To identify these features, we can decompose the given series into its components, however we are left with a random component. In finance these random components have correlation – for analysts it is a goal to understand these random component correlations and a reason why these trading algorithms can become very complex. One of the most basic time series models is **Random Walk**, however Random Walk is unable to identify this autocorrelation. The goal is to produce a model that can explain autocorrelation.

Stationarity in Time Series:

The mean of a given time series x(t) is:

$$\mathbf{E(x(t)) = \mu(t)}$$

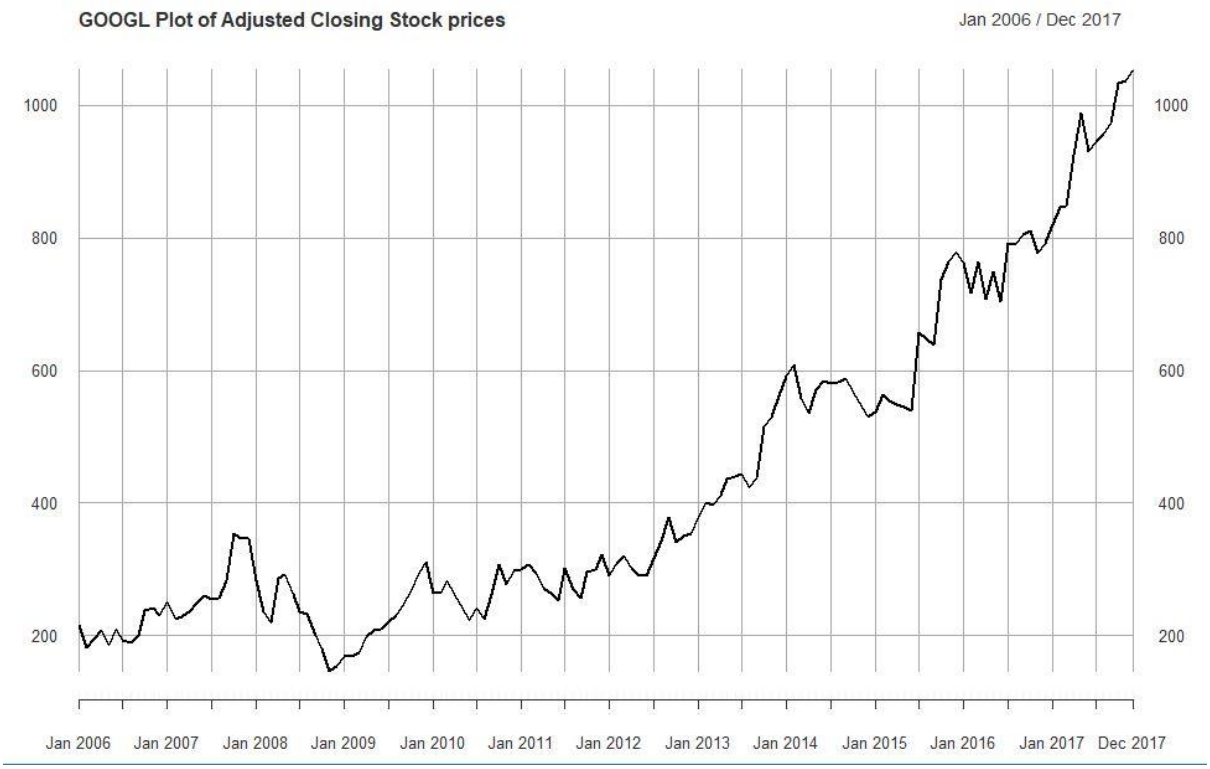
A time series is **stationary** (as far as the mean is concerned) if the mean and variance are constant e.g. white noise. The mean and variance are completely independent of time.

$$\mathbf{Mean\ is\ constant:\ \mu(t) = \mu}$$

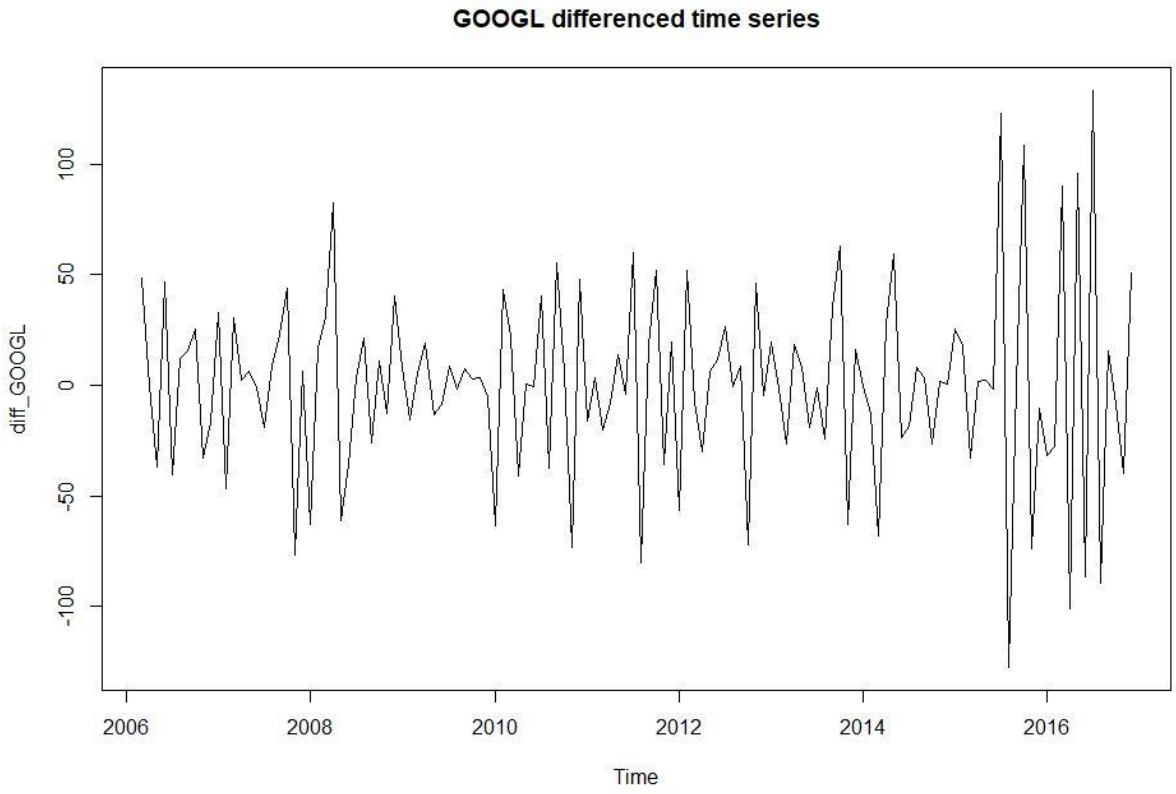
$$\mathbf{Variance\ is\ constant:\ \sigma(t)^2 = \sigma^2}$$

There are many time series where the above will not be true i.e. observations may be correlated, so we may over or underestimate the actual variance. This is what happens when there is autocorrelation.

Non-stationary time series example:



Stationary time series example:



White Noise

White noise is a stationary process as the mean is constant (i.e. zero) and the variance is constant – they don't change over time. White noise if very important when dealing with residuals but also when dealing with random walk as the random walk model relies heavily on white noise.

Random Walk

We can define a random walk with white noise:

$$\mathbf{x(t) = x(t\text{-}1) + w(t)}$$

the actual value of a given stock = previous stock value + random fluctuation (white noise)

This is the discrete model of a random walk.

The previous value of a stock can be recursively decomposed into a previous value and a random term. E.g. x(t-1) can be x(t-2) + w(t-1) and so forth – hence Random walk is the sum of white noise series.

Important: the mean of the random walk process is 0 but the autocorrelation is not constant – it depends on time which means random walk is not a stationary process.

So, we have the following:

Random walk = a deterministic part (the previous value) + a stochastic part (white noise).

To improve on the random walk model, we could:

1. Consider more observations in the past – the Autoregressive model (AR)
2. Add more white noise terms – instead of manipulating the deterministic part we will manipulate the white noise part (in order to eliminate all serial correlation) – the Moving Average model (MA)

Autoregressive Model (AR)

A linear model the AR model is the generalisation of the random walk. In this model future observations are said to have a linear relationship with past observations plus a random error and constant term.

$$x_t = a_1 x_{t-1} + \ldots + a_p x_{t-p} + w_t$$

An autoregressive model of order p, where p represents the number of lags considered in the model, e.g. if p = 2 we consider 2 previous observations and so on. Note that if a = 1 this is equivalent to the random walk equation. We had earlier that a random walk is not stationary thus the AR model is also not always stationary – thus it is unable to explain autocorrelation all the time. Examination of the PACF plot can help in choosing the order of an AR model.

Moving Average model (MA)

With the MA model instead of considering more and more past observations we will consider more and more stochastic white noise parts or past errors, i.e. we add more white noise. MA models combine white noise terms from the past – this is called a q ordered moving average model:

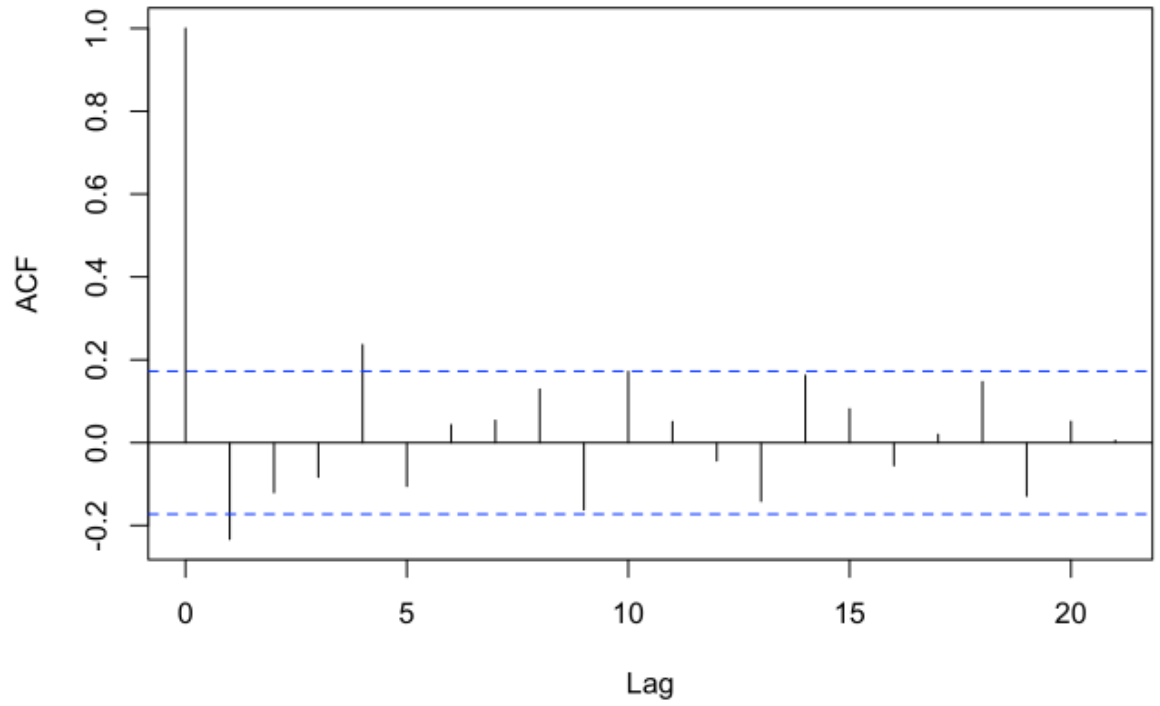
$$x_t = \beta_1 x_{t-1} + \ldots + \beta_q x_{t-q} + w_t$$

Also, the mean of this model (x) is 0 – the mean of white noise is 0.

Note with MA models there can be serial correlation – this is a fundamental feature of MA models. If there are some autocorrelations as far as the first q lags are concerned, then it is advised to use a MA model with order q. To determine the order q of an MA model you would use the ACF plot.

Correlogram – ACF & PACF plots

The correlogram (or auto correlation ACF plot) is a plot of the autocorrelations versus the time lags – it shows the serial correlation present in the data that changes over time.



The time lags are on the x axis (bottom) and the autocorrelation function (ACF) are on the y axis. Important to note that ACF doesn't have any dimensions as stated earlier, correlation itself is defined as being dimensionless.

Note that when the lag = 0 the ACF will always = 1, not matter what time series you are considering.

When the lag = 1 we shift the time series 1 step to the right, for example:

X1	X2	X3	X4	X5	X6	
	X1	X2	X3	X4	X5	X6

When the lag = 2 we shift the time series 2 steps to the right and so on.

The correlogram will show us what time lags have some serial correlation. For example, in the above ACF plot there is some positive correlation when the lag = 1 and when lag = 4. The blue dotted lines are the 95% confidence level – if the lags are within these blue lines then with 95% confidence we can say there is no correlation and vice versa.

The correlogram also helps us determine the AR and MA orders as follows:

	ACF plot	**PACF plot**
AR	Has geometric decay to zero	And any correlation spikes drop suddenly after 1+ lags **then this indicates an AR of order P**
MA	Correlation spikes drop suddenly after 1+ lags	And has geometric decay to zero **then this indicates an MA model of order q**

Potential mixed order of AR and MA	Significant spikes at random lags	Significant spikes at random lags

ARMA models

We can go one step further and combine AR and MA models to potentially create a better model of order p and q. This model was constructed in 1970 by Box and Jenkins. Again, we can use the auto correlation plots to find the p and q parameters. We can also utilise **Akaike Information Criterion (AIC**). AIC is an estimator of the relative quality of a model. However, AIC doesn't inform you about the model, it informs you relative to other models so if all possible models fit poorly AIC won't warn you. It penalises models that are overfit (penalises models with lots of parameters) but rewards goodness of fit. The lower the result for AIC the better.

Please see the included R file "ARMA 1". Here I simulate an ARMA model with order p of 3 (0.4, -0.2, 0.6) and order q of 2 (0.6, -0.4).

I use a for loop and AIC to determine the p and q parameters and orders. The model with the lowest AIC value is chosen. Examining the residuals using the ACF plot no correlations are present and the Ljung Box test confirms same – the ARMA (3,2) model is working well.

Also, please see the included R file "ARMA STOCK". The code determines that (3,3) is the best ARMA model to use to model the S&P 500 stock time series however there is still some significant serial correlation (using the Ljung box test), therefore for this particular time series the ARMA model is not good enough to explain the correlation.

Transformation of a non-stationary time series

We can take a non-stationary process (random walk) and perform a transformation process such as differencing y(t) – y(t-1) and transform it into a stationary process (white noise). This transformation step is very important as non-stationary data is unpredictable and cannot be modelled so if we can transform the data to stationary we can then start to build models.

Please see included the R file "random walk APPL" where I difference a time series of Apple stock data and show that it has been transformed from non-stationary to stationary – the residuals are like white noise.

ARIMA models

AR and MA models work when applied only on stationary time series. Therefore an ARIMA model combines AR and MA models with differencing (d) which allows us to transform a non-stationary time series to stationary. ARIMA models use differencing repeatedly d times in order to reduce a non-stationary series to end up with a stationary series i.e. try and eliminate all serial correlation. Non-stationary data is unpredictable and cannot be modelled hence the transformation into a stationary process as this allows us to create models that enable prediction.

1st Order Differencing Formula:

$$y'(t) = y(t) - y(t-1)$$

Please see included the R file "ARIMA". Here I simulate an ARIMA (1,1,1) model that is non-stationary. I then transform the time series by differencing once. I then run a manual ARIMA model of (1,1,1) and am able to produce similar coefficients within the confidence levels. Analysis of the residuals prove no autocorrelation present.

CHAPTER 3: METHODOLOGY & IMPLEMENTATION

Model Design & Methodology

Throughout this project a CRISP-DM methodology of data analysis was followed and the following framework describes the main methods that were followed in examining and applying time series models through to accuracy of prediction:

Examine and understand the data

- Initial data exploration - visualise the data – are the any obvious patterns or observations?

Data Preparation

- Decompose the time series - check for trend and any seasonality present in the data
- Check for stationarity - if necessary stablize the data by applying a transformation method e.g. differencing (unit root test) or logarithms

Modelling Stage

- Correlogram/ACF & PACF plots of the differenced data (if differencing is necessary) to try and determine possible models
- Estimate the parameters of model and fit the model

Evaluation Stage

- Gauge the model fit using AIC for example

Deployment/Reevaluation

- Predict the future values
- Check the accuracy of the model- predict vs actual

As stated earlier the main time series model I implemented was the ARIMA model. What follows is a review of my findings.

I chose stocks from unrelated industries as follows:

Company name	Industry	Ticker symbol	Data set
Alphabet (Google parent company)	Technology	GOOGL	11 years of (monthly) stock data from Jan 2006 to Dec 2017
Nike	Consumer goods – Apparel Footwear & Accessories	NKE	11 years of (monthly) stock data from Jan 2006 to Dec 2017
Exxon Mobil Corporation	Basic Materials – Oil & Gas	XOM	11 years of (monthly) stock

			data from Jan 2006 to Dec 2017

Examine and understand the data

My stock data is gathered from the Yahoo Finance website using the getSymbols function of the quantmod package. Yahoo Finance provides a number of stock prices on their site:

Open	The opening stock price that day/month
High	The highest stock price that day/month
Low	The lowest stock price that day/month
Close	The closing stock price for that day/month (this price is adjusted for splits)
Adjusted Close	The adjusted close price adjusted for both dividends and splits

I used the adjusted close price on a monthly basis for my data set for each companies stock as this price includes that bit more information in it than the other options – the more information the better!

Plot of Google stock from Jan 2006 to Dec 2017:

Initial observations: a clear upwards trend, no apparent seasonal components, and the variance seems to be random over time.

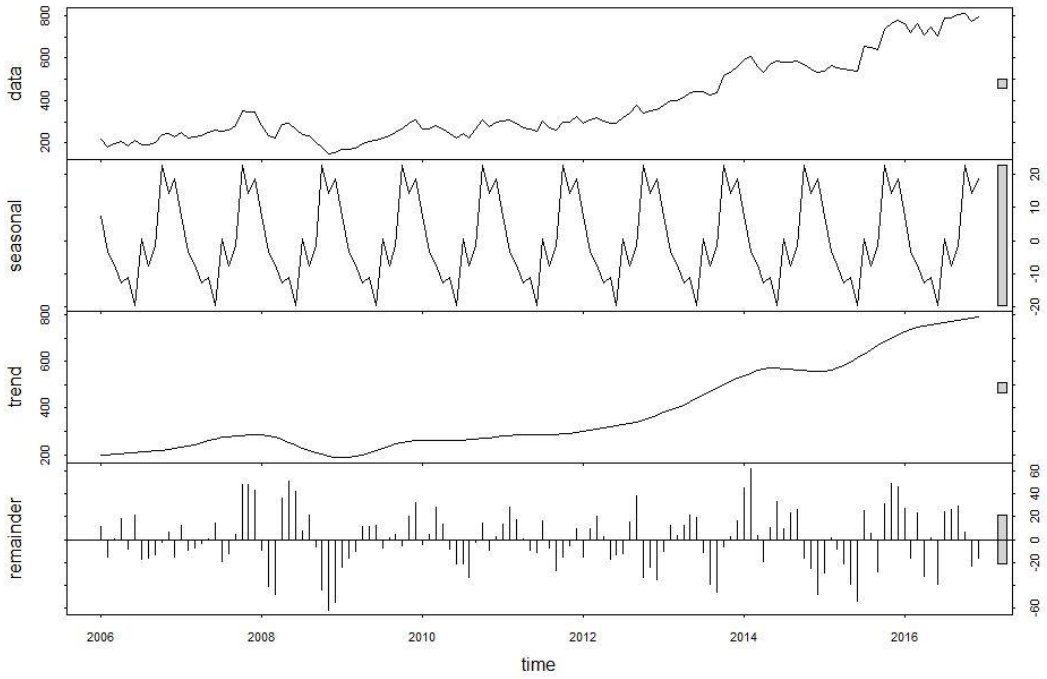
Data Preparation

Split the dataset:

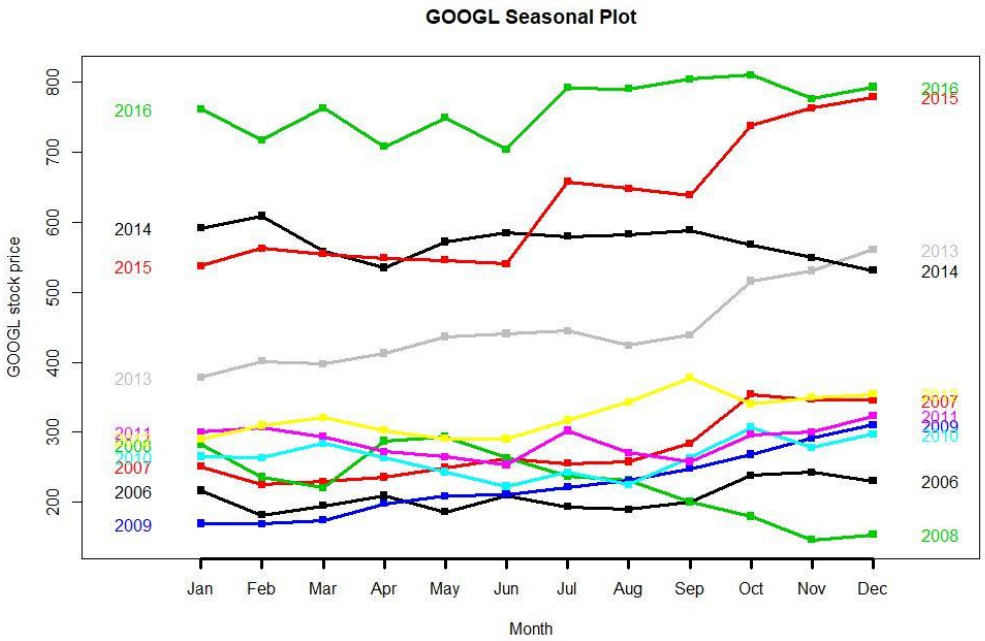
I had 11 years worth of monthly data from Jan 2006 to Dec 2017 but the goal was to apply an ARIMA model and predict the monthly stock prices for the full year of 2017 for each company stock. Therefore, I split the dataset (training and held out) and held out the 2017 data to use as the actuals at the evaluation stage.

Decompose the time series:

This plot helps to further analyse trend, seasonality, and residuals.



Here we can see the upward trend in the data again from 2006 to end 2016 but also there seems to be a seasonal element to the data across the years. However looking at a core seasonal plot this is hard to make out:



We are looking to see are there any patterns from month to month that resemble each other across the years. To me no obvious patterns are visible other than the fact that most years the price finishes higher than it started.

Check for correlations and stationarity:

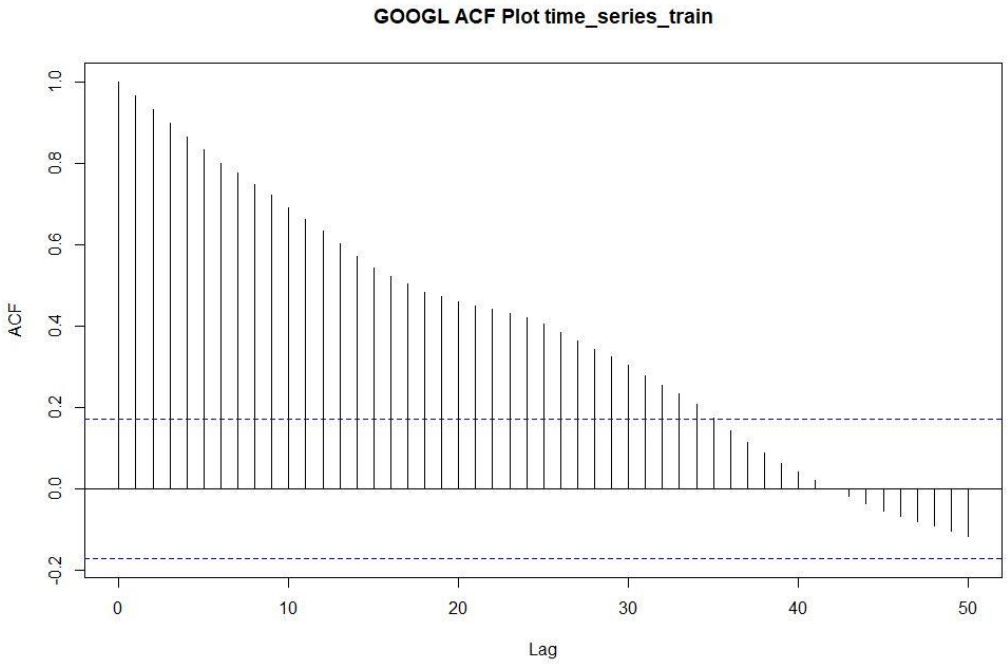
The Ljung box test results for the Google dataset resulted in a pvalue < 0.05 meaning there is serial correlation and the time series is non-stationary:

```
# Box-Ljung test
#
# data:  times_series
# X-squared = 2062.6, df = 45, p-value < 2.2e-16
# With this low p value we can determine that there is serial correlation
# and the time series is non-stationary
```

Next the Dickey-Fuller test resulted in a pvalue > 0.05 again indicating non-stationarity.

```
# Augmented Dickey-Fuller Test
#
# data:  times_series
# Dickey-Fuller = -0.57887, Lag order = 5, p-value = 0.9771
# alternative hypothesis: stationary
# With a high P value like this we will have to difference the time series
```

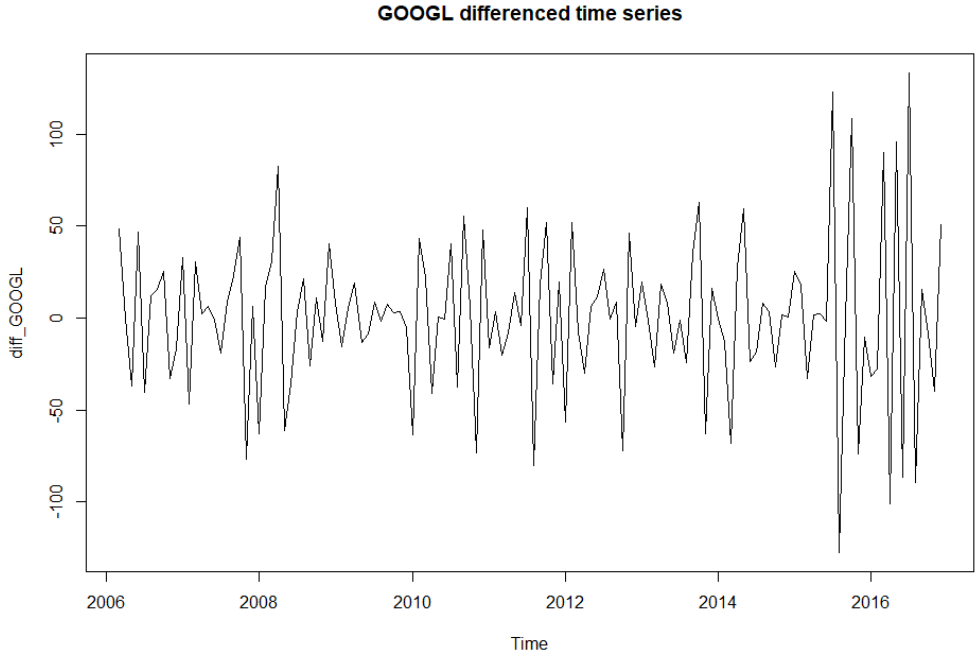
Furthermore the ACF & PACF plots/correlogram were implemented to check correlations and stationarity. Below is the ACF plot for Google training data set. There are many ACF values above the 95% condifence levels slowly decay to zero which indicates these lags have serial correlation but also the time series is indeed non-stationary.



Modelling Stage

Once a time series is stationary further tests can be performed and new correlations/properties discovered. I stablized the time series by transforming the dataset using differencing. As stated earlier differencing tries to eliminate all serial correlation and reduce the series to a stationary one.

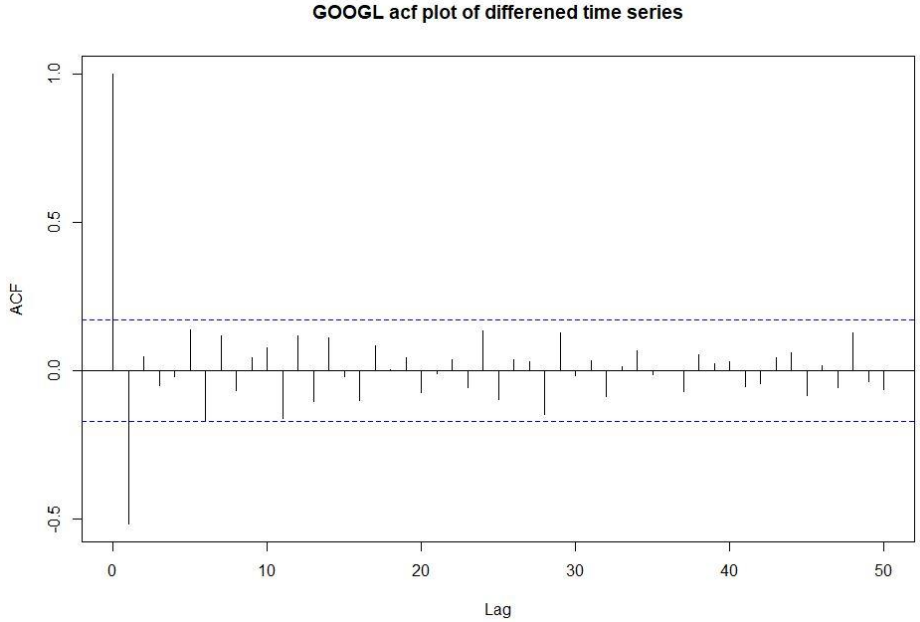
Below is the the differenced plot for Google, note differencing was performed twice in order to approximate the mean as close to zero as possible.

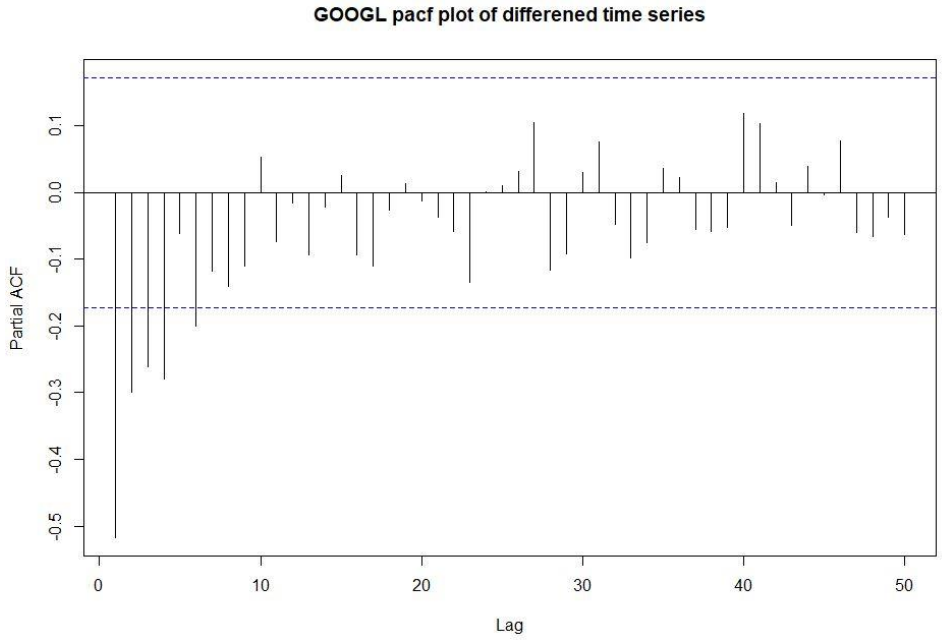


The Dickey Fuller test is then used to determine stationarity. The pvalue was < 0.05 meaning we now have a stationary time series.

```
# Augmented Dickey-Fuller Test
#
# data:  diff_GOOGL
# Dickey-Fuller = -20.028, Lag order = 0, p-value = 0.01
# alternative hypothesis: stationary
```

To help determine the order of our ARIMA model I plotted ACF and PACF plots of the differenced time series. From the ACF plot there is an initial spike at lag 1 but no more correlations after that. From the PACF plot the correlations seems to decay slowly. This would indicate an MA (1) model.





Therefore as I have differenced the time series twice already and have established an MA(1) order, our final ARIMA model to test on the training dataset is of the order 0,2,1.

I used 2 methods to fit ARIMA models: the arima and auto.arima functions from the forecast package in R. This was done to help verify the model order I had established when analysing the ACF/PACF plots as auto.arima calculates the best ARIMA model order automatically.

Below are the summary results from the manual arima fit and auto.arima functions on the training dataset.

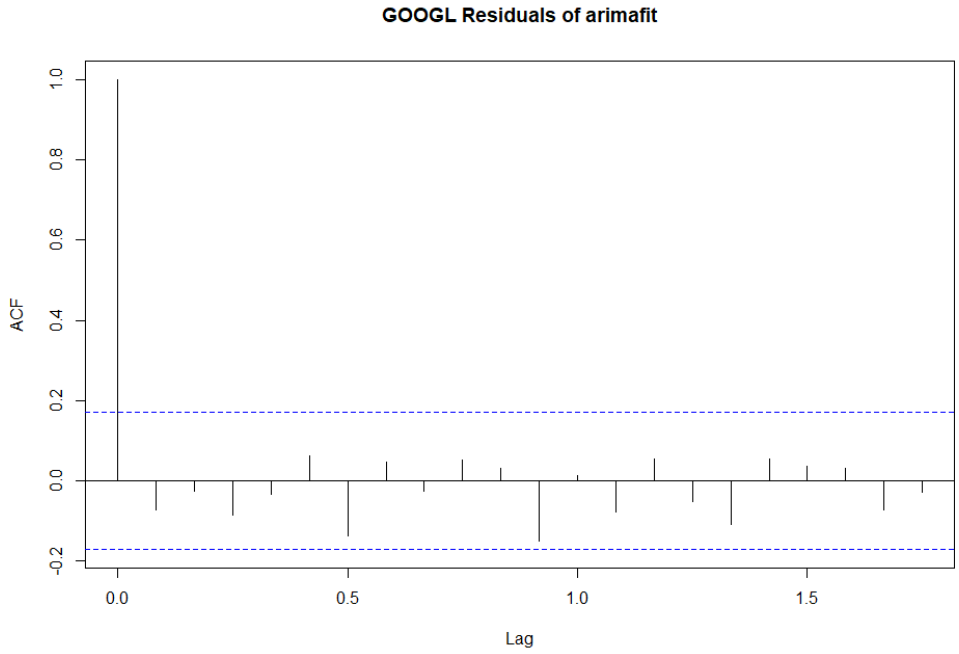
```
# Series: times_series_train
# ARIMA(0,2,1)
#
# Coefficients:
#           ma1
#       -0.9903
# s.e.   0.0350
#
# sigma^2 estimated as 870.2:  log likelihood=-625.86
# AIC=1255.72   AICc=1255.82   BIC=1261.46
#
# Training set error measures:
#                     ME     RMSE      MAE        MPE     MAPE      MASE         ACF1
# Training set 2.874344 29.16214 21.66404 0.5036992 6.370402 0.2725742 -0.07194367
```

```
# Series: times_series_train
# ARIMA(0,1,0) with drift
#
# Coefficients:
#        drift
#       4.3962
# s.e.  2.5480
#
# sigma^2 estimated as 857:  log likelihood=-627.73
# AIC=1259.46   AICc=1259.55   BIC=1265.21
#
# Training set error measures:
#                       ME     RMSE      MAE         MPE     MAPE      MASE         ACF1
# Training set 0.001607199 29.05203 21.42977 -0.7596913 6.337574 0.2696266 -0.07067459
```

Interestingly auto.arima has calculated the best model to be of order 0,1,0 i.e. no AR or MA order which is effectively a random walk model. However looking at the AIC and BIC results the manual arima fit produces slightly better results. On the other hand the error measures are slightly lower for the auto.arima model. Based on the AIC result I chose the manual arima fit of order 0,2,1.

Evaluation Stage

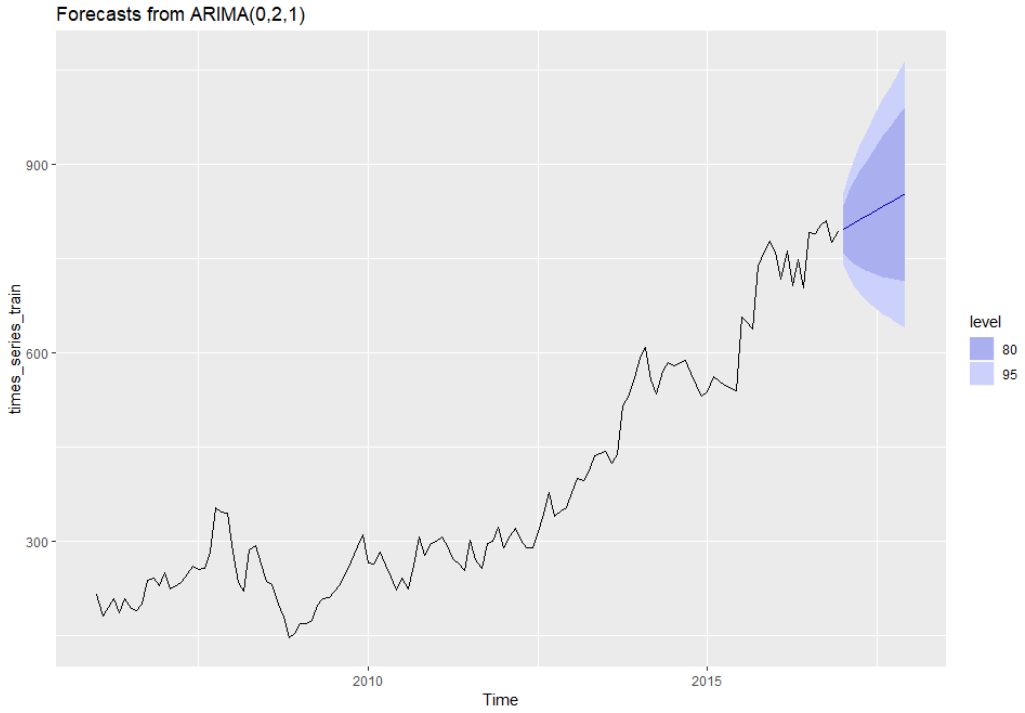
To evaluate the model I inspected the residuals. The Ljung box returned a pvalue of 0.9347 and the ACF plot below which both indicate stationarity, white noise (no correlations which is good) and goodness of fit.



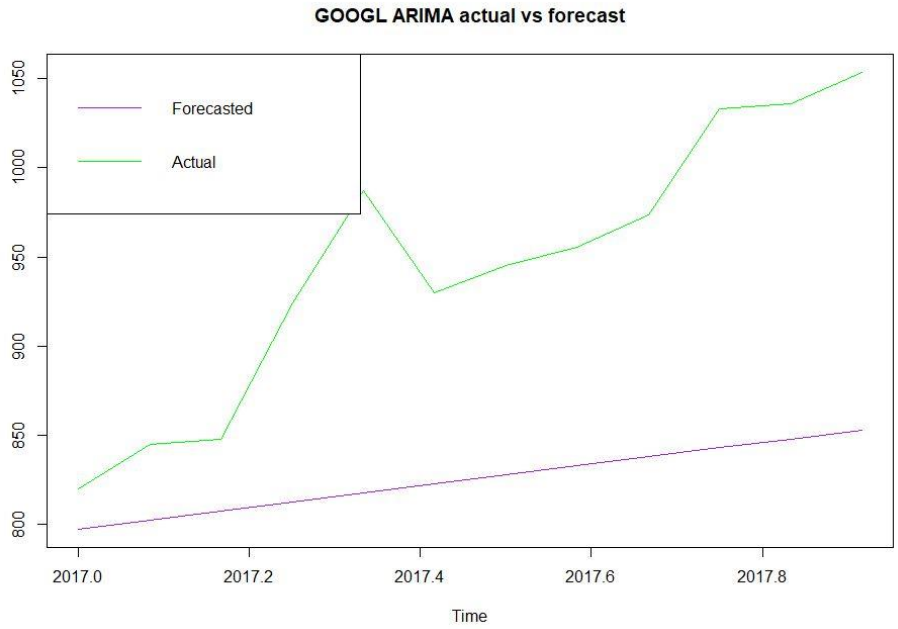
However at this evaluation stage it is important to iterate again if the Ljung box test is < 0.05 (unfavourable) or there are correlations present when plotting the residuals you should go back and repeat the fitting and diagnostics again.

Deployment of models

Happy with the model I set about using it to predict the stock prices for the 12 months of 2017. Below is a plot of the forecasted prices for the Google stock. The result being a fairly linear upward trend, also the confidence levels cast a wide net which is not ideal.



Below is a close up plotting forecasted vs actual stock prices.



I used the accuracy function to generate measures of accuracy between predicted and actual, the results for Google as follows:

	ME	RMSE	MAE	MPE	MAPE
ARIMA	120.6854	134.0408	120.6854	12.3418	12.3418

Alternative Forecast Methods

For each stock chosen I also performed some alternative forecast methods also available via the forecast package in R as a comparison to the popular ARIMA method. Please see the Appendix for the forecasts these models produced.

The following models were applied:

- Neural Network time series forecasts – nnetar function
- Exponential Smoothing state space model – ets function
- TBATS model (exponential smoothing state space model with Box-Cox transformations) – tbats function

The accuracy measures are outlined below along with the initial ARIMA method.

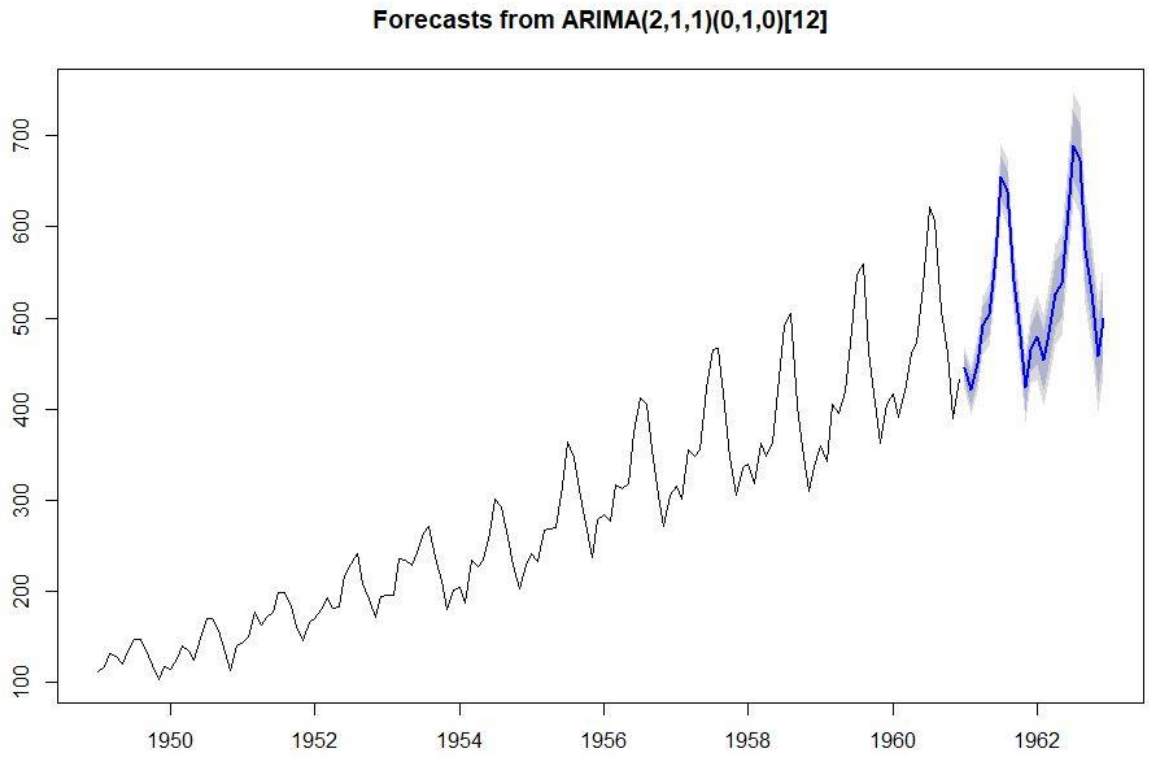
	ME	RMSE	MAE	MPE	MAPE
ARIMA	120.6854	134.0408	120.6854	12.3418	12.3418
Neural Net	164.9068	181.3654	164.9068	16.89588	16.89588
Expo Smooth	153.4918	170.5175	153.4918	15.6931	15.6931
TBATS	153.2536	170.3032	153.2536	15.66776	15.66776

Critical Evaluation

As we can see the ARIMA model performs a lot better when comparing the predict stock prices vs the actual using Mean Error (ME) as the main accuracy measure. Exponential smoothing and TBATS have nearly equal ME results while the neural network model is further off. When comparing the forecast plots for all models the ARIMA model seems to grasp the upward trend in the time series the best, with exponential and TBATS being flat forecasts and neural network having a slightly downward forecast.

However, what seem like "flat" or mostly linear forecasts may actually be reasonable predictions. You have to consider the prediction confidence intervals (80% and 90%) i.e. the variation in predicted stock prices that are shown on the forecast plots. When comparing the plots again for each model on Google it's plain to see the ARIMA model indeed as the narrowest prediction intervals. TBATs is second followed by exponential smoothing (note I wasn't able to produce intervals for the neural network model in R).

As a comparison I applied an ARIMA model on the well-known air passenger's dataset available in R (see air passengers appendix). The ARIMA model works really well at modelling the time series and predicting future passenger levels. As you can see the confidence intervals hug the prediction forecasts tightly providing excellent results.



Analysing the same forecast plots for the Nike and Exxon Mobil stock data, overall the ARIMA model performs best with the narrowest confidence bands. However, with the variation and skewness of these intervals being so large the predictions seem very naïve.

Over the 3 stock datasets the most popular ARIMA model was of the order (0,1,0) which is effectively a random walk model i.e. a non-stationary process. This is effectively telling me that the stock price is a random amount that is uncorrelated with past observations.

AR and MA models assume that the volatility in the series is constant which is not the case. These simple models don't care for volatility clustering and are limited by the fact they assume a linear form to time series. Yes, stocks can have a linear trend over long periods of time perhaps but as history has shown there can be times of huge economic change and shocks to the markets. From my initial research it seems to me that the ARIMA family of time series modelling is better suited to time series that contain more regular seasonal patterns as shown by the air passengers' dataset e.g. annual temperatures or business sales figures.

In this research I utilised monthly data but I also tried using daily stock data in the hope that with more information comes better predictions. However, this was not the case as I produced similar results as to when using monthly data. It does seem the ARIMA model can't quite cope with the randomness and volatility of stock market prices. Essentially, we need a model that takes this volatility into account.

GARCH Models

To extend my research I investigated GARCH models. GARCH models can be applied to series that do not have any trends or seasonal effects, so a better strategy would be to apply an ARIMA model first to account for these effects and then apply a GARCH model in order to explain volatility clustering.

Please find the included R file "GARCH ARIMA GOOGL". Here I apply an ARIMA model to the 11 years' worth of Google stock price data followed by a GARCH model to try and explain volatility. Again, I examined the ACF plots of the GARCH residuals and discovered approx. white noise however on plotting the squared residuals there appears to be random correlations still present. The GARCH

model doesn't seem to be able to explain all of the volatility once again but does a pretty good job. For more information on this work please see the GARCH APPENDIX.

PLEASE NOTE FOR ALL OF MY R CODE PLEASE SEE THE INCLUDED R FILES.

CHAPTER 4: CONCLUSION & FUTURE WORK

I set out to see how accurate and reliable modern time series analysis models can be in predicting the future, and in particular stock market prices. From the research carried out and application of various models it is fair to say I wouldn't be relying on any of my model predictions in the real world.

The ARIMA models I implemented on Google, Nike and Exxon Mobil stocks, while they seemed to reasonably understand the general trend of the time series the confidence levels leave a lot to be desired. It is clear linear ARIMA models have huge problems dealing with volatile time series and seem to be much more suited to seasonal patterned datasets. The benefit of using ARIMA models is their ease of application and simplicity, plus their accuracy on seasonal/cyclical data.

By merging ARIMA and GARCH models together I feel I've only scratched the surface when it comes to building more complex time series models. I hope that my work done here can be used as a benchmark and spur others to further investigate more complex time series models.

However, it is important to state that my dataset was stock prices – no other information, data points or features. This research was a form of technical analysis in that all the information is contained in the data but in the real world of stock markets anything is possible. Machine algorithms may be able to spot trends in stock prices but I believe there is a long way to go before machines can incorporate all the outside variables that influence the rise and fall of stock markets. A combination of machine models, experience and expertise in this field and I'm sure one could make a tidy little profit, well maybe.

REFERENCES

Yule, G.U. 1927. *On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers.* Philosophical Transactions of the Royal Society of London Series A, 226, 267– 98.

Robert H. Shumway and David S. Stoffer 2011, *Time Series Analysis and its Applications with R 3rd Edition*, Springer Text.

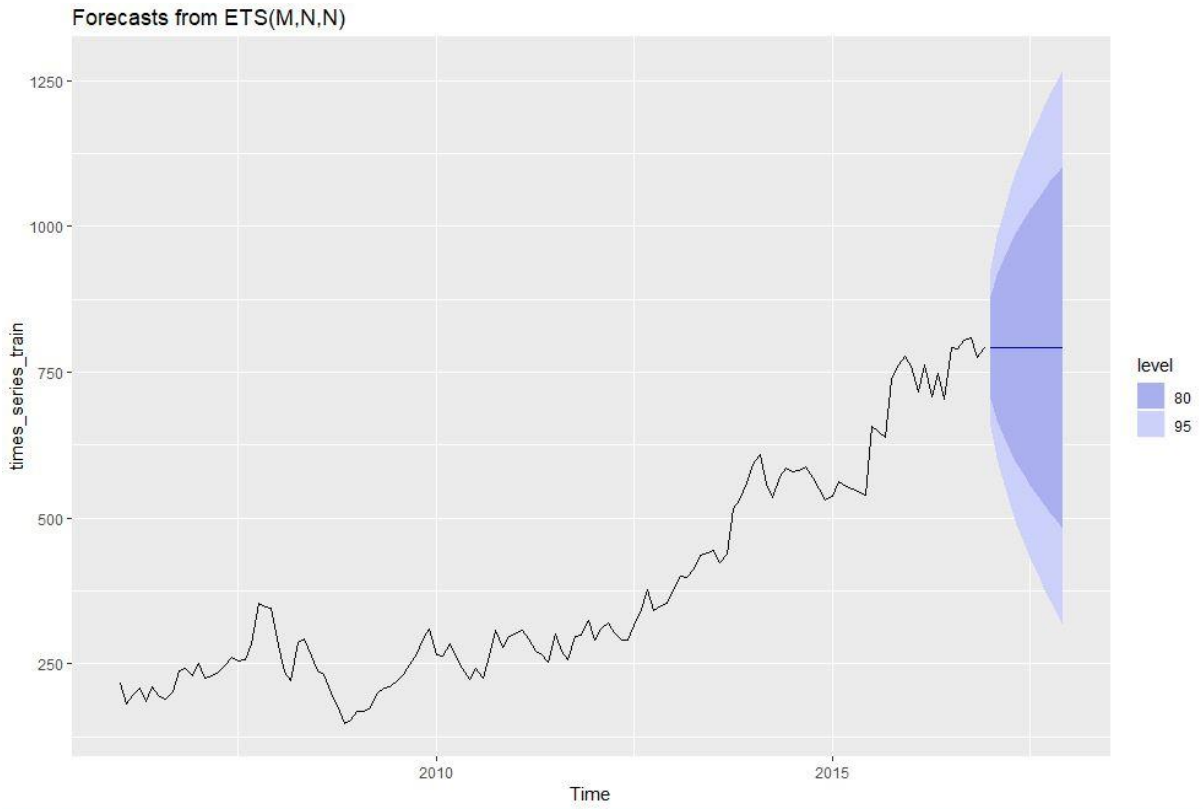
Ruey S. Tsay 2002, *Analysis of Financial Time Series*, Wiley & Sons.

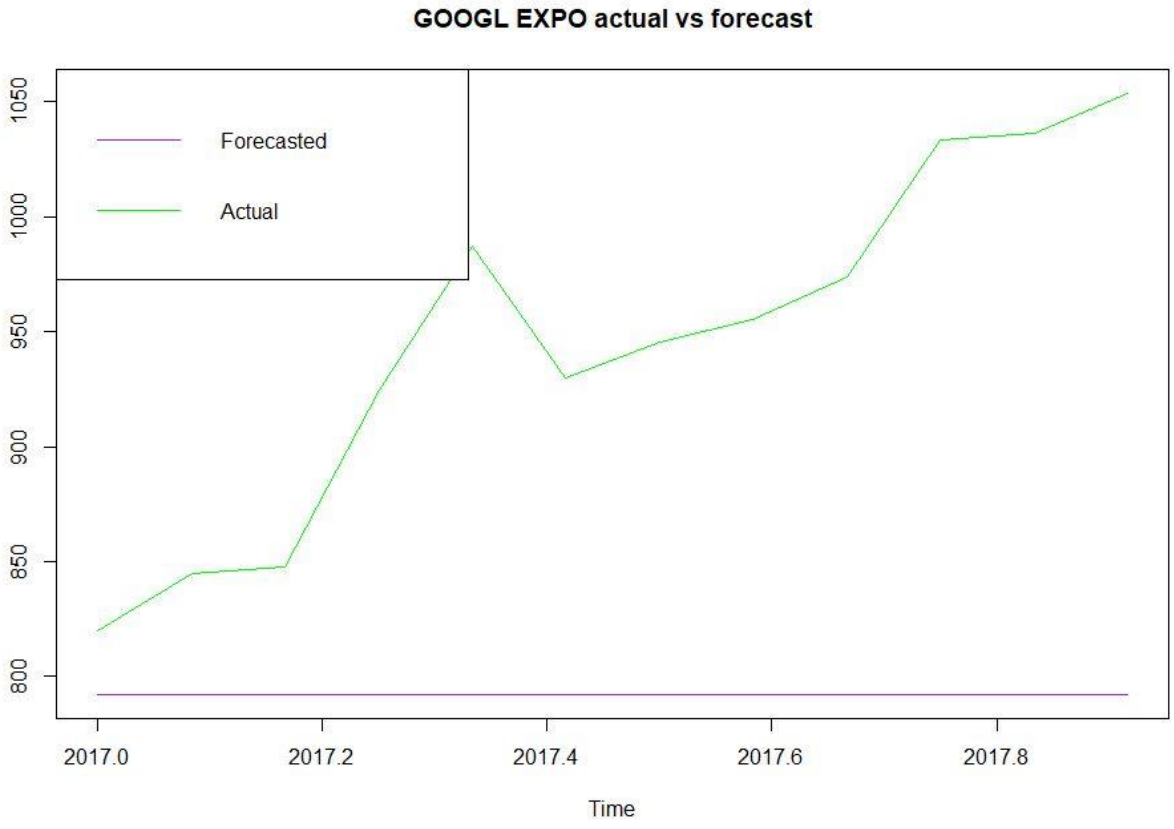
Rob J Hyndman and George Athanasopoulos, *Forecasting: Principles and Practice*, accessed 1st June June 2018, <https://otexts.org/fpp2/>

APPENDIX

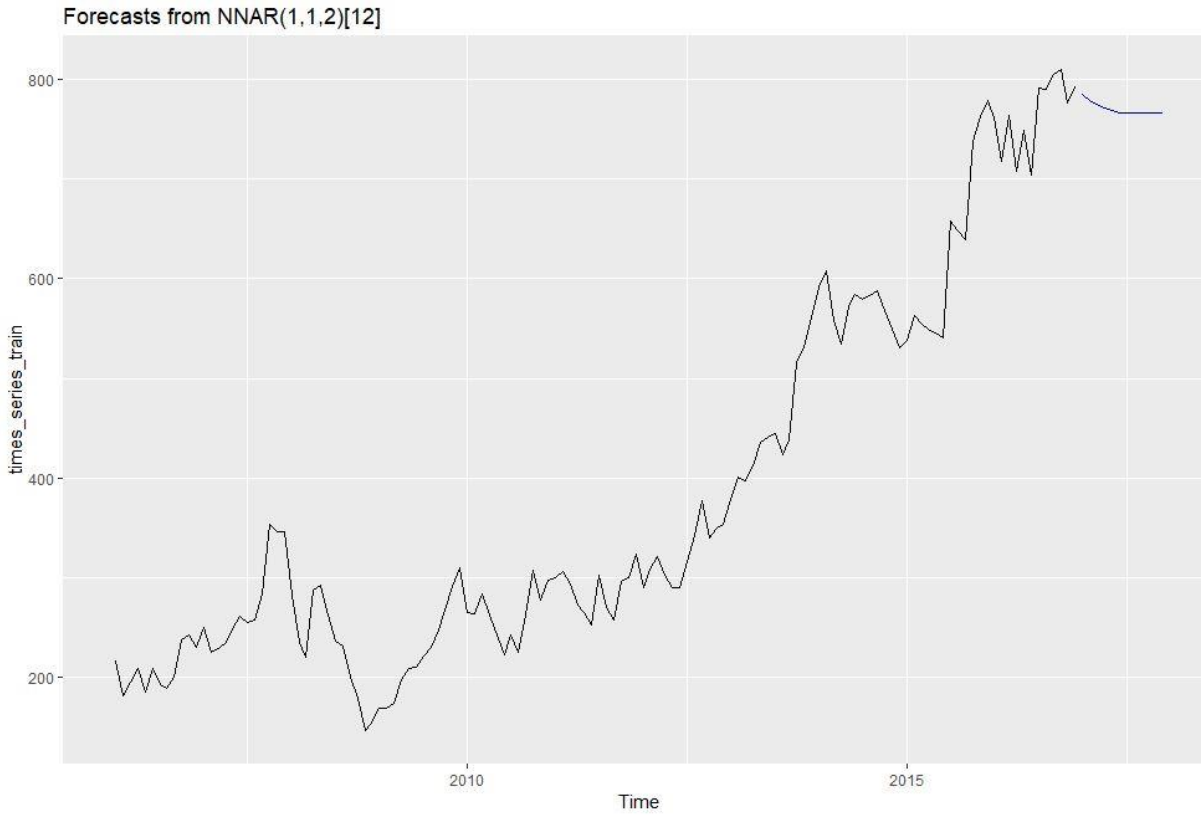
Google Appendix – plots alternative forecast methods

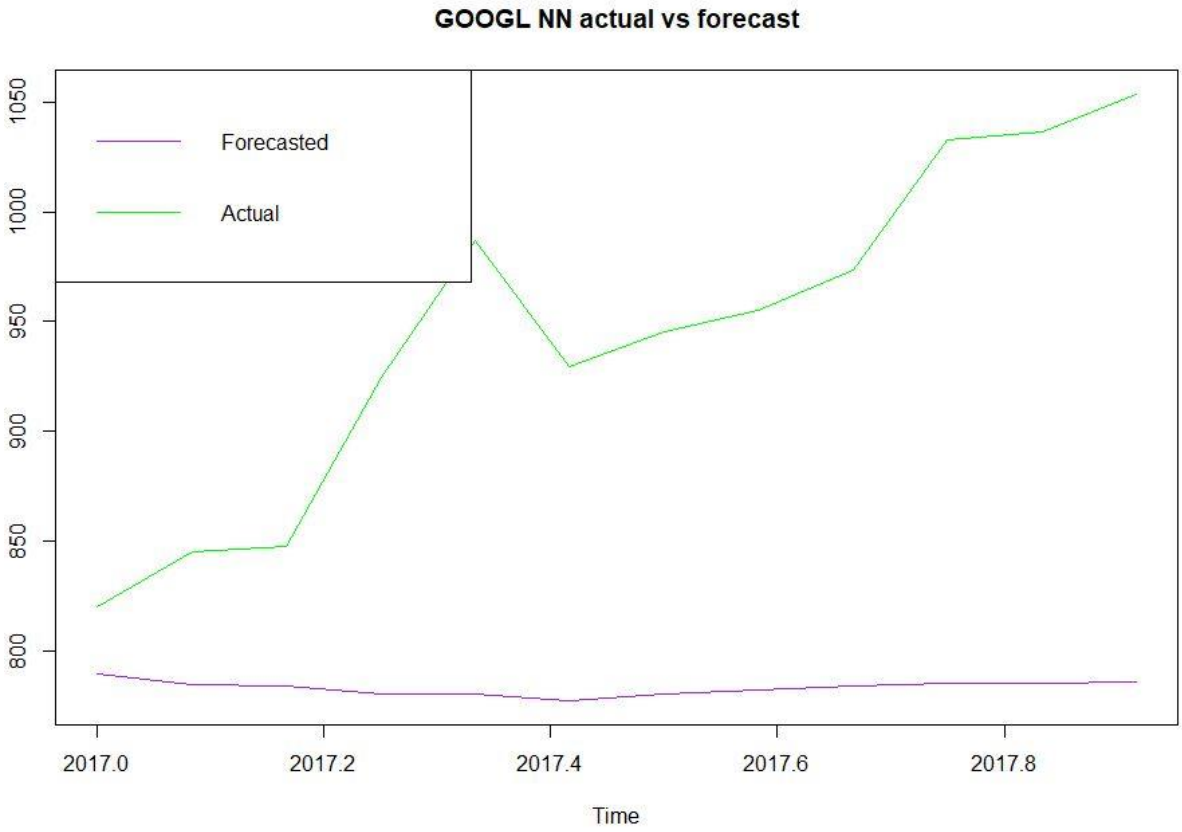
Exponential Smoothing State Space Model.



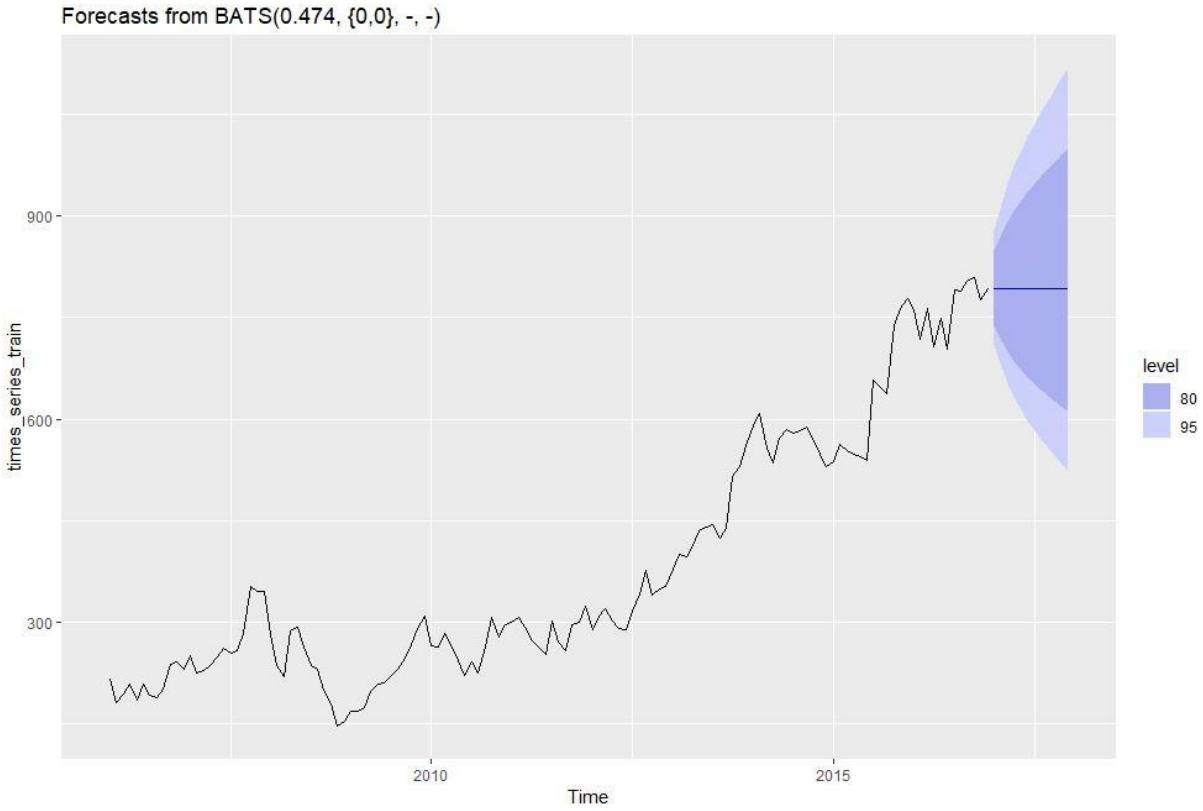


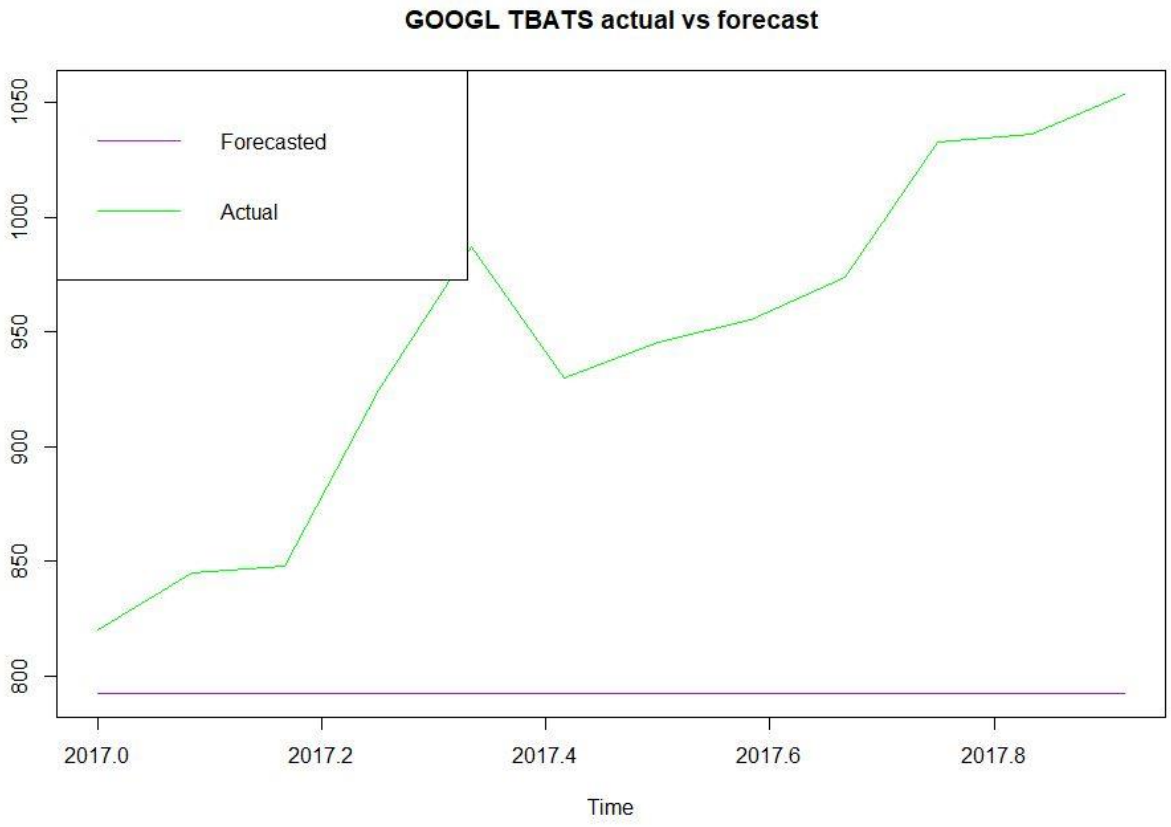
Neural Network: This is a feed-forward neural network with a single hidden layer and lagged inputs for forecasting univariate time series.





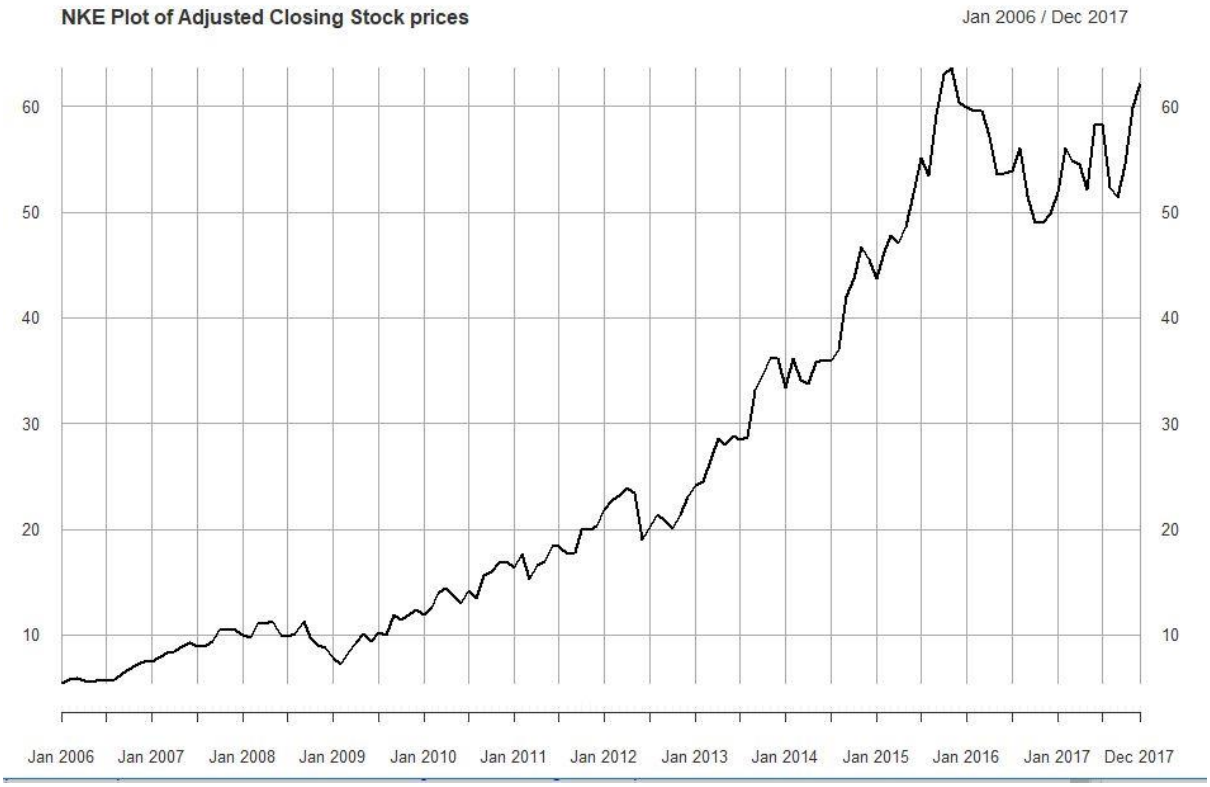
TBATS model is an exponential smoothing state space model with Box-Cox transformation, ARMA errors, Trend and Seasonal components.





NIKE APPENDIX

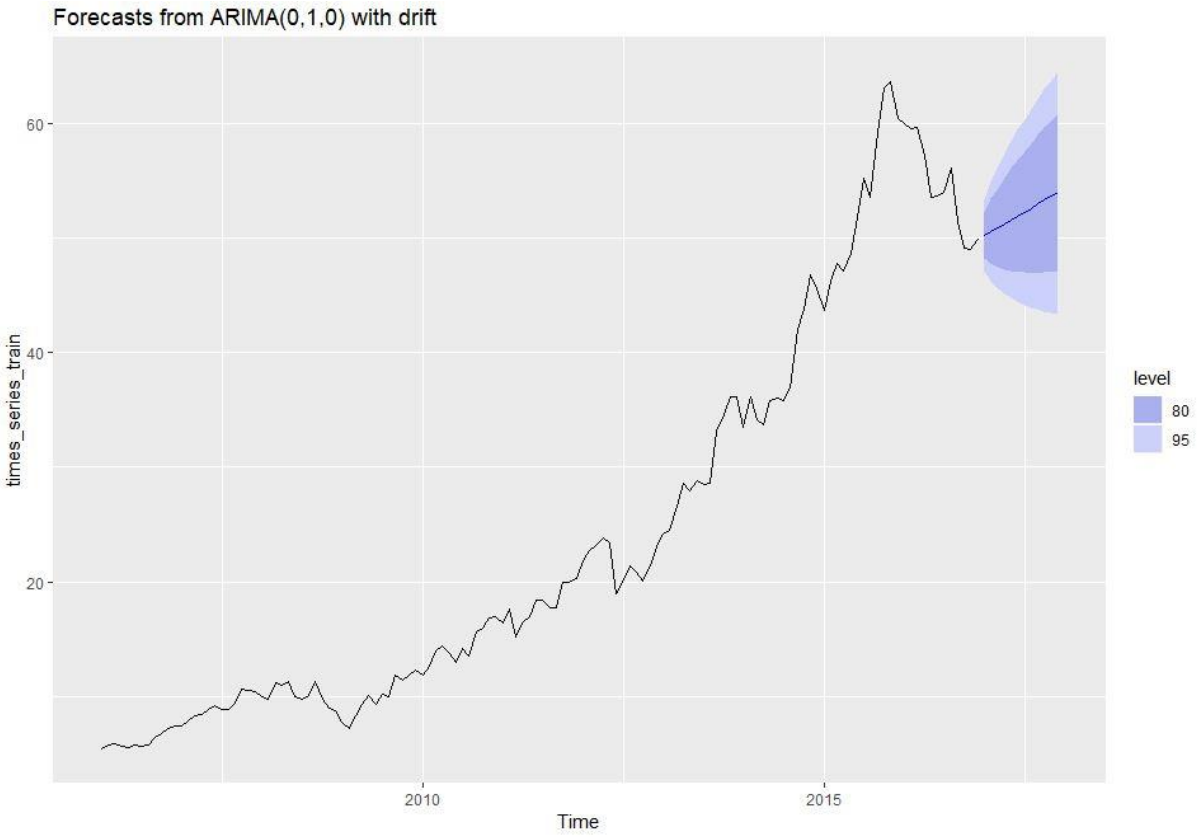
Original time series plot – 2006 to 2017

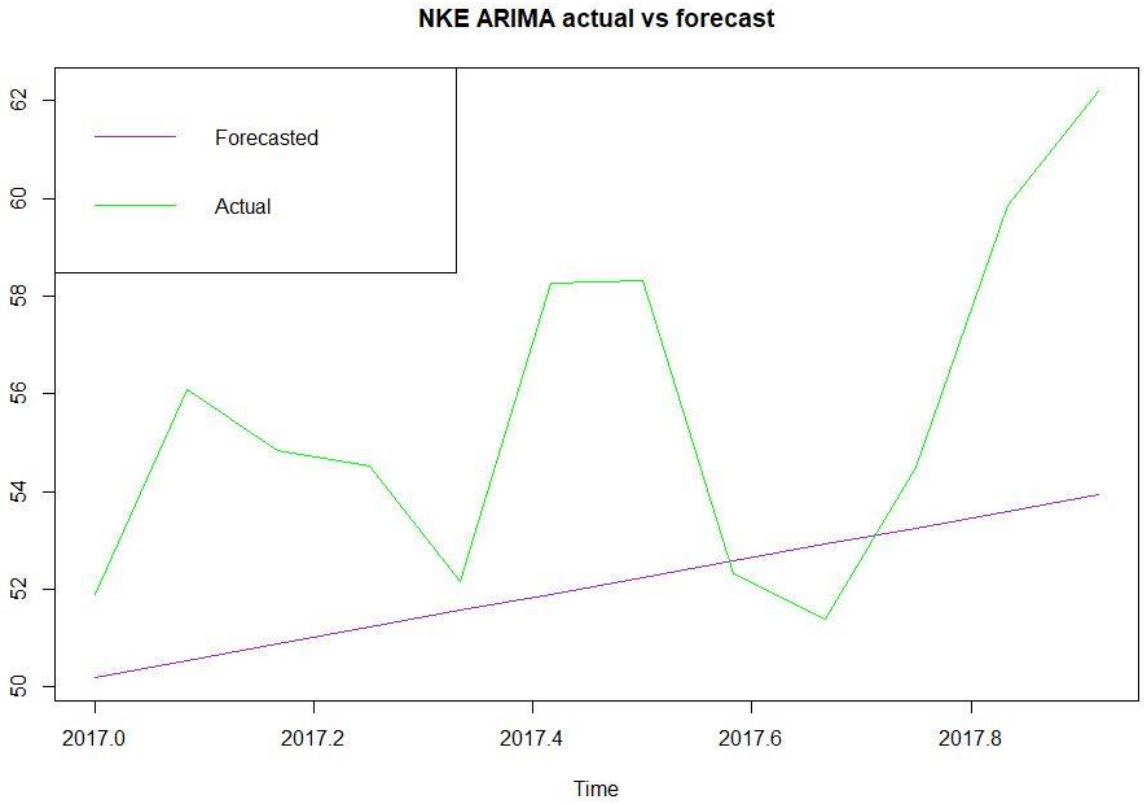


Accuracy measures for each model on the Nike dataset:

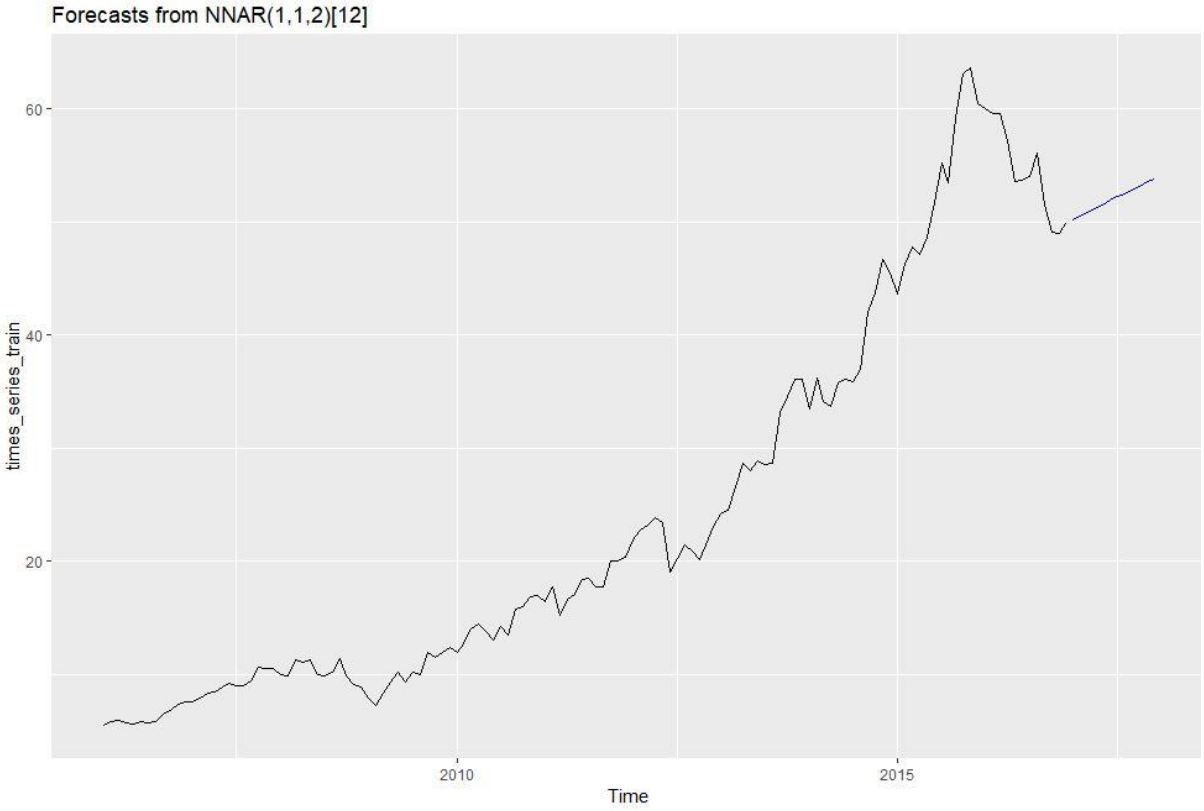
	ME	RMSE	MAE	MPE	MAPE
ARIMA	3.45405	4.554767	3.753655	5.945422	6.527056
Neural Net	3.539793	4.610833	3.78805	6.100947	6.583585
Expo Smooth	4.14554	5.138482	4.244956	7.178449	7.371944
TBATS	2.064153	3.608438	3.203401	3.47079	5.656869

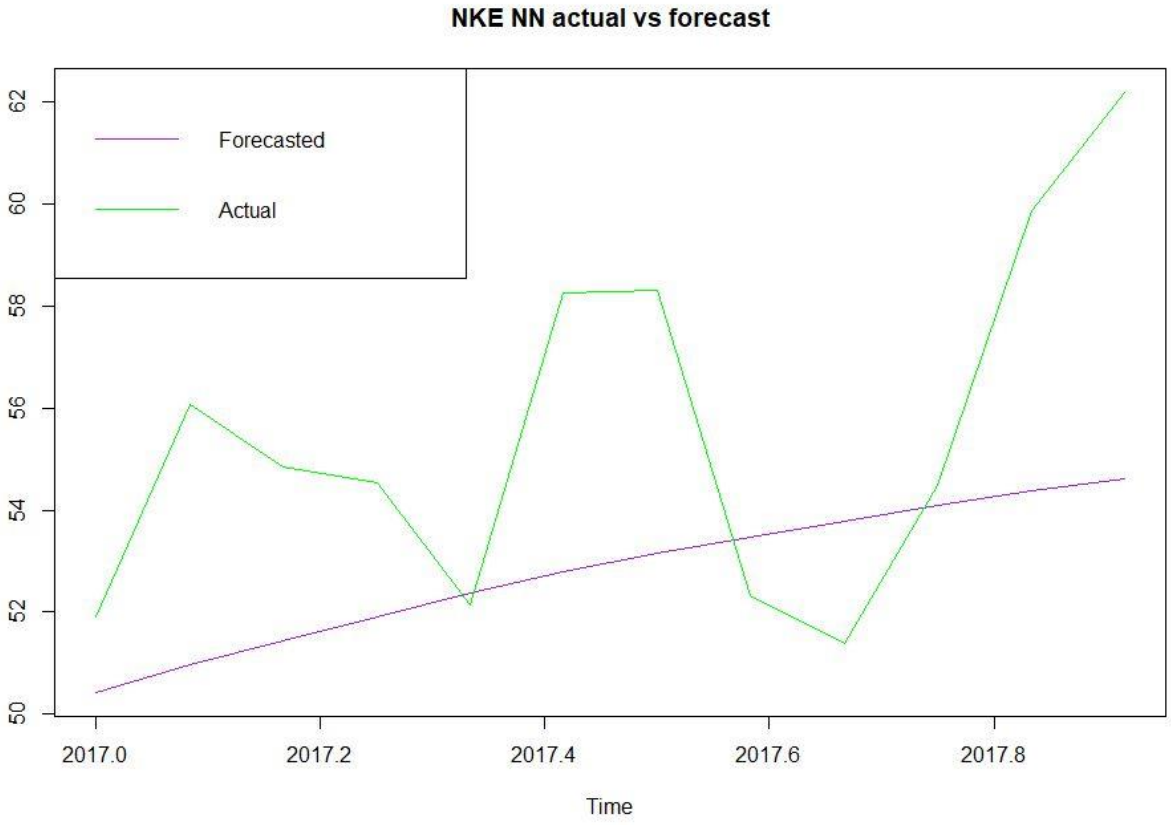
ARIMA forecast plots:


Forecasts from ARIMA(0,1,0) with drift
times_series_train
60
40
20
2010
2015
Time
level
80
95

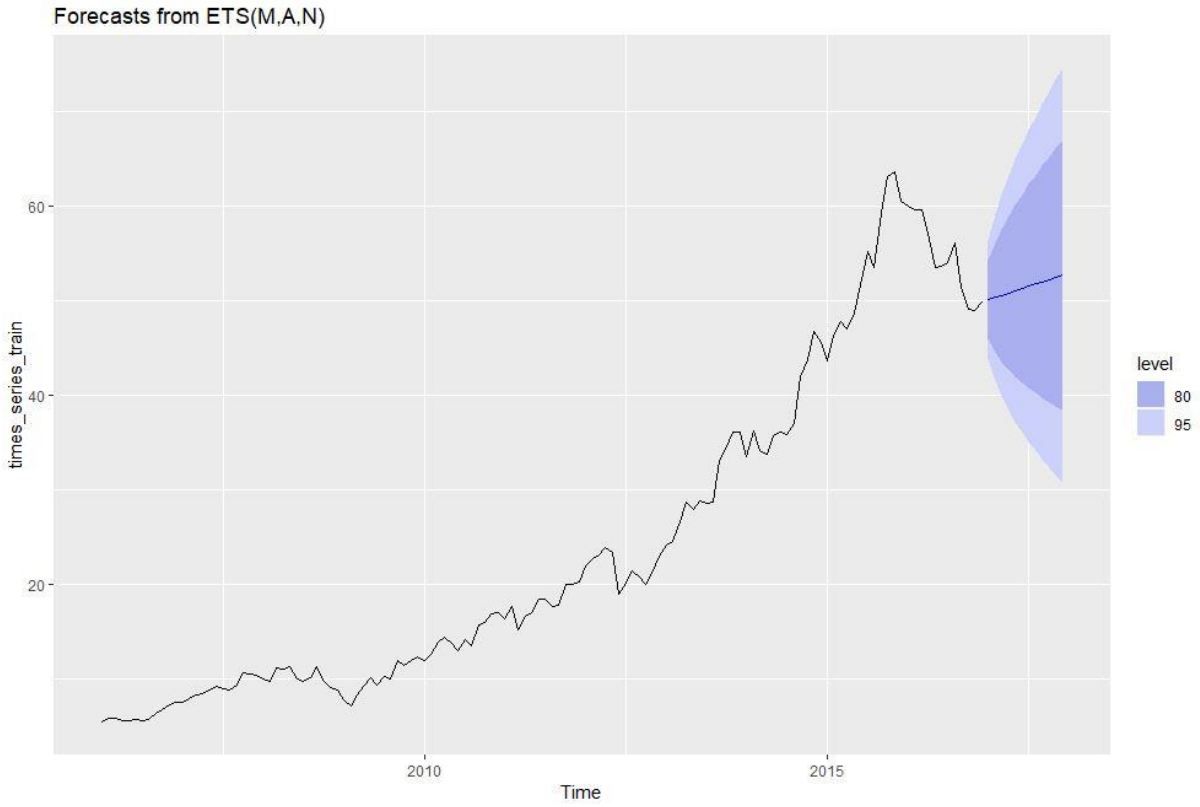

NKE ARIMA actual vs forecast
Forecasted
Actual
62
60
58
56
54
52
50
2017.0
2017.2
2017.4
2017.6
2017.8
Time

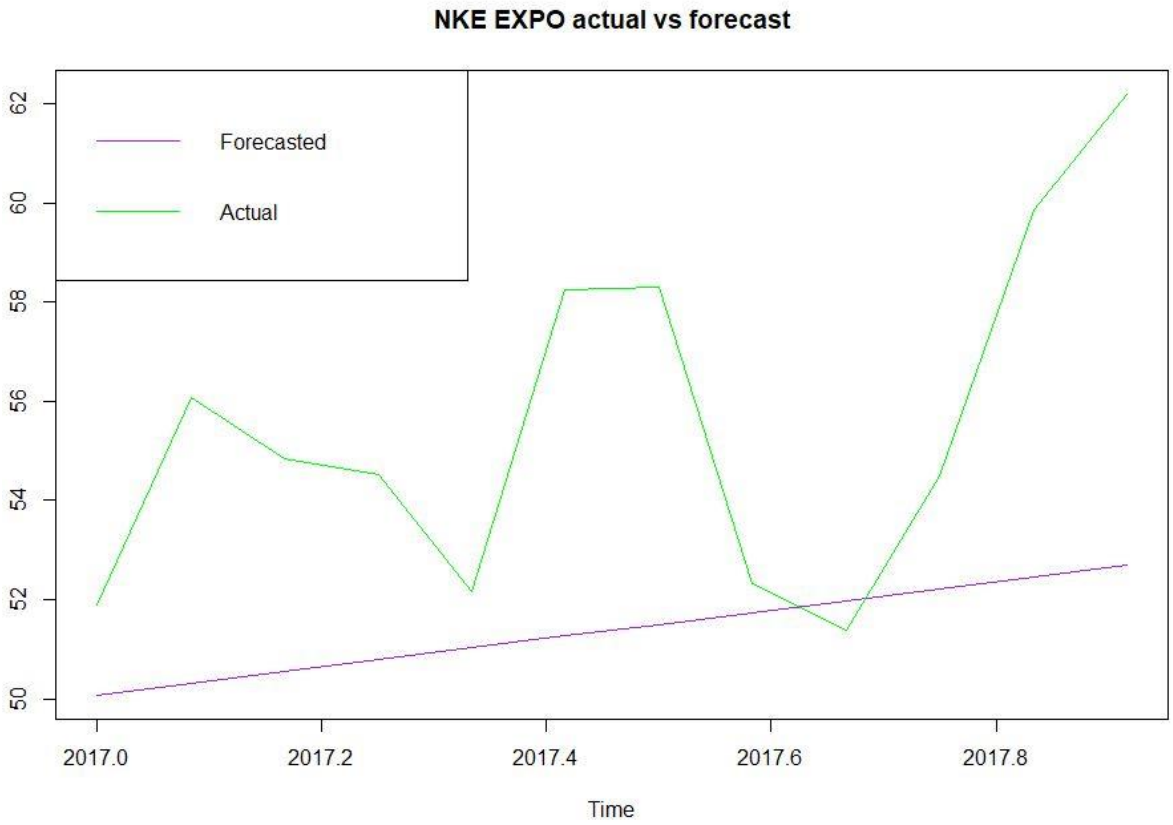
Neural Network:



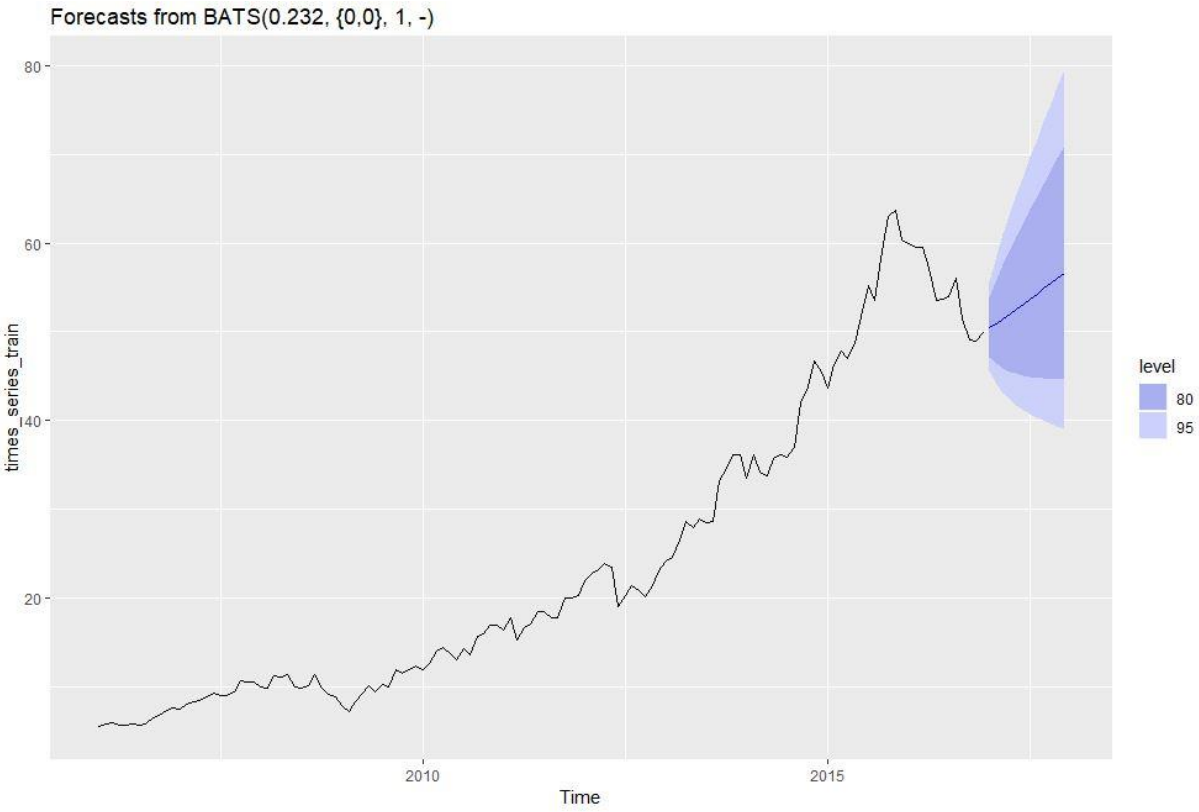


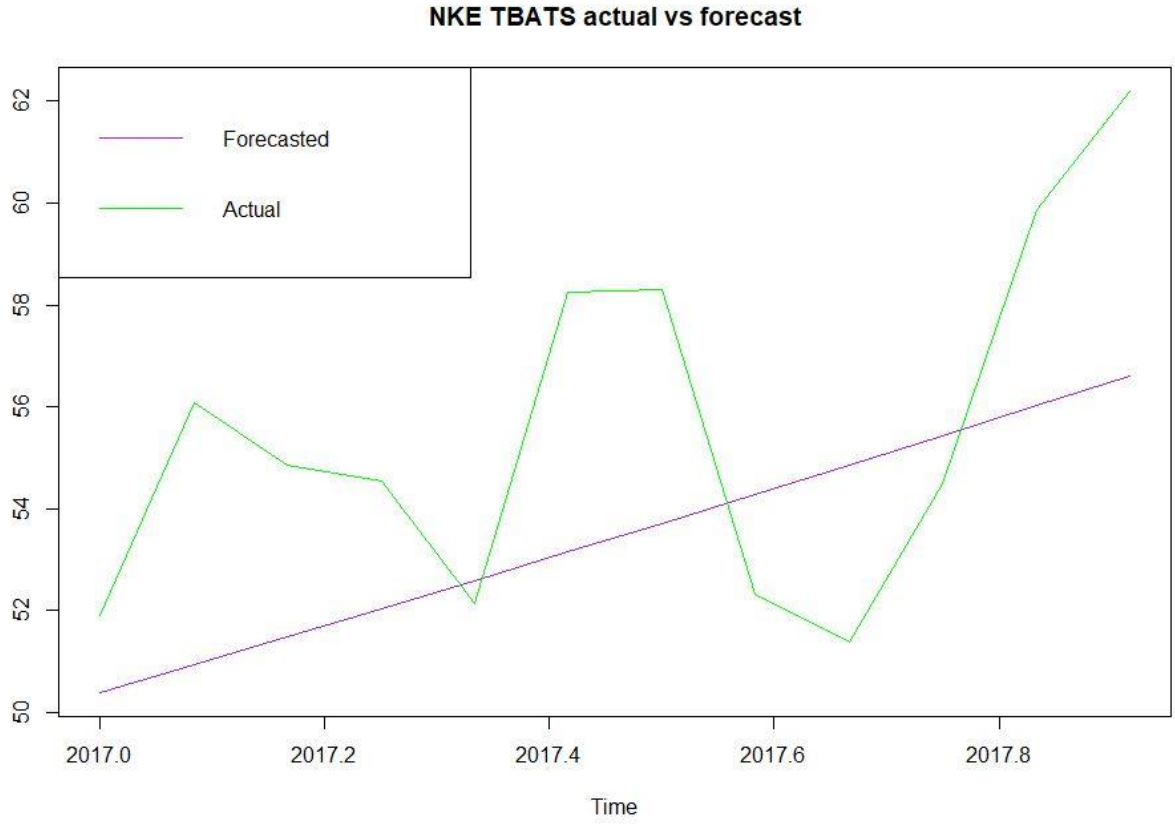
Exponential Smoothing:





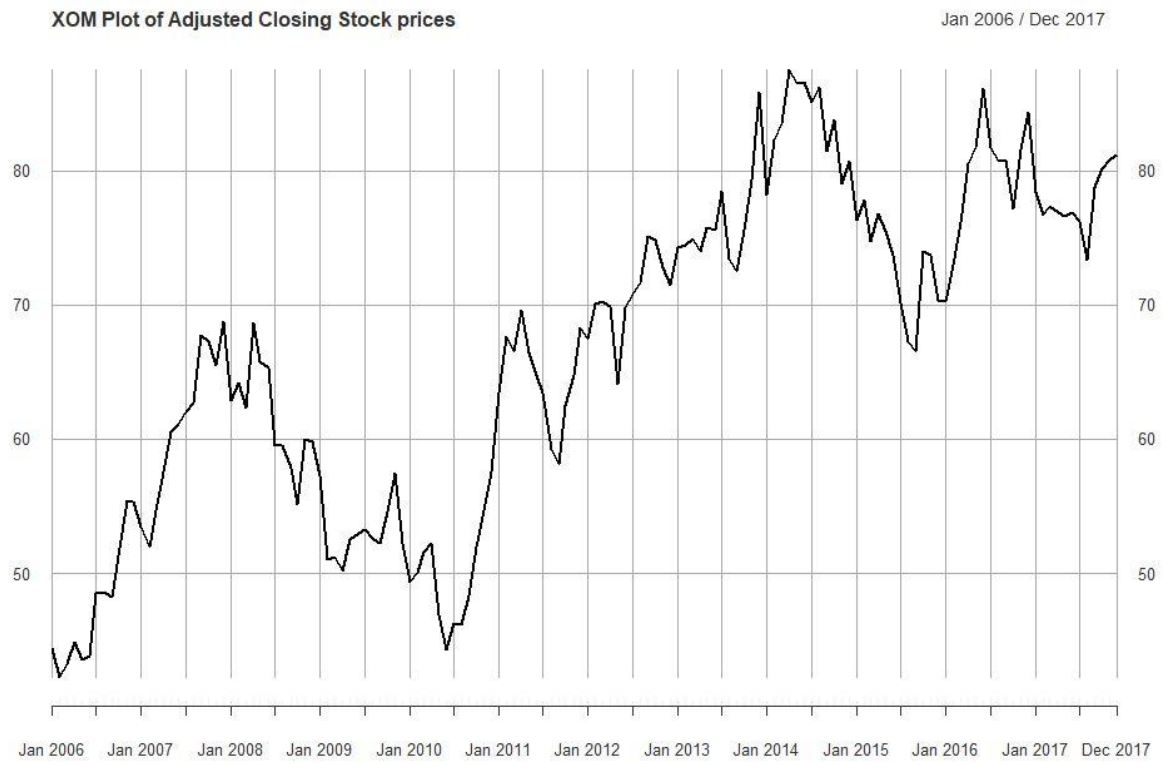
TBATS:


Forecasts from BATS(0.232, {0,0}, 1, -)
times_series_train
80
60
40
20
2010
2015
Time
level
80
95


NKE TBATS actual vs forecast
Forecasted
Actual
62
60
58
56
54
52
50
2017.0
2017.2
2017.4
2017.6
2017.8
Time

EXXON MOBIL APPENDIX

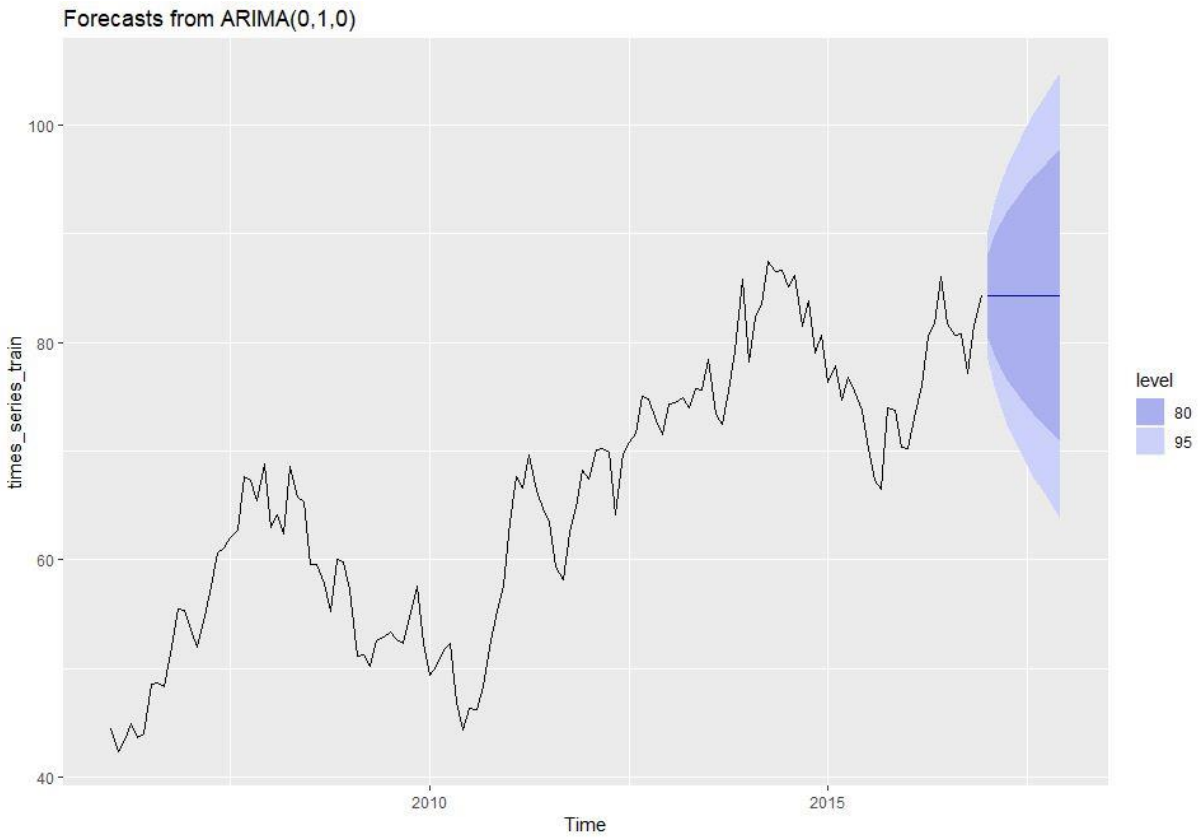
Original time series plot – 2006 to 2017

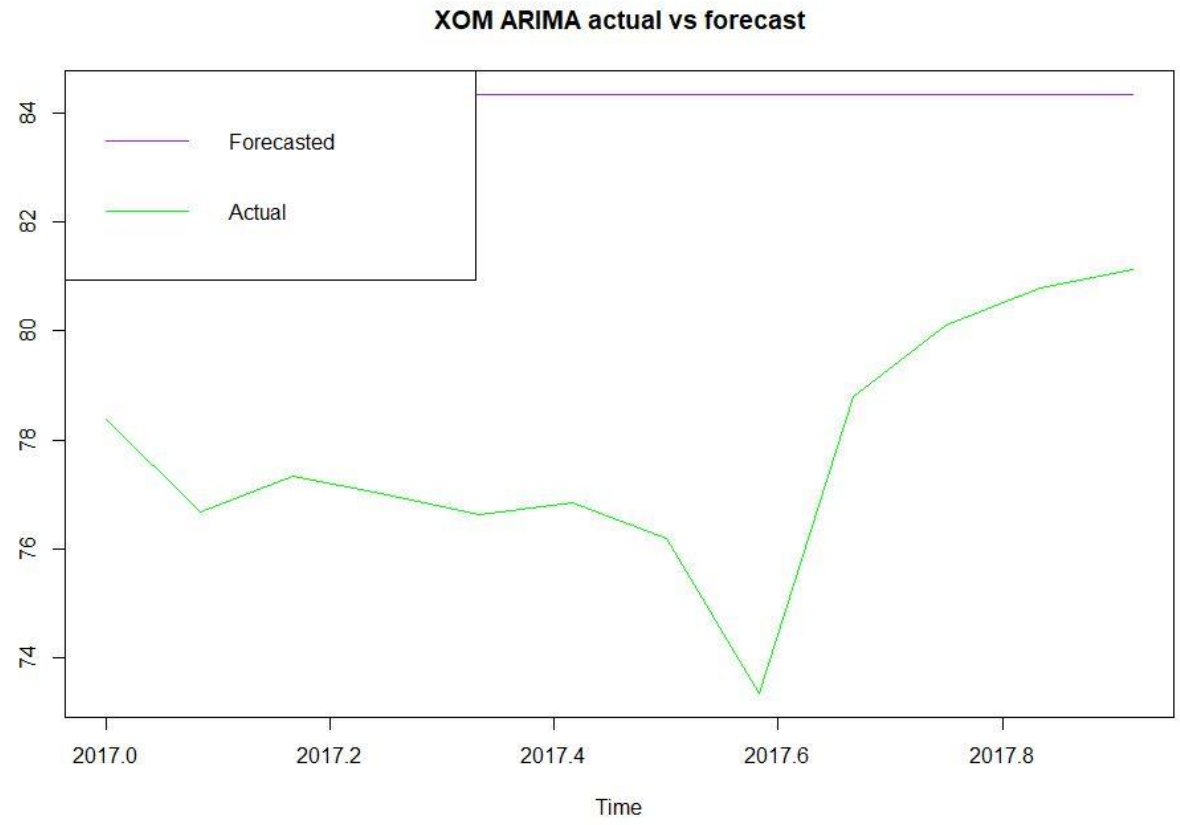


Accuracy measures for each model on the Exxon Mobil dataset:

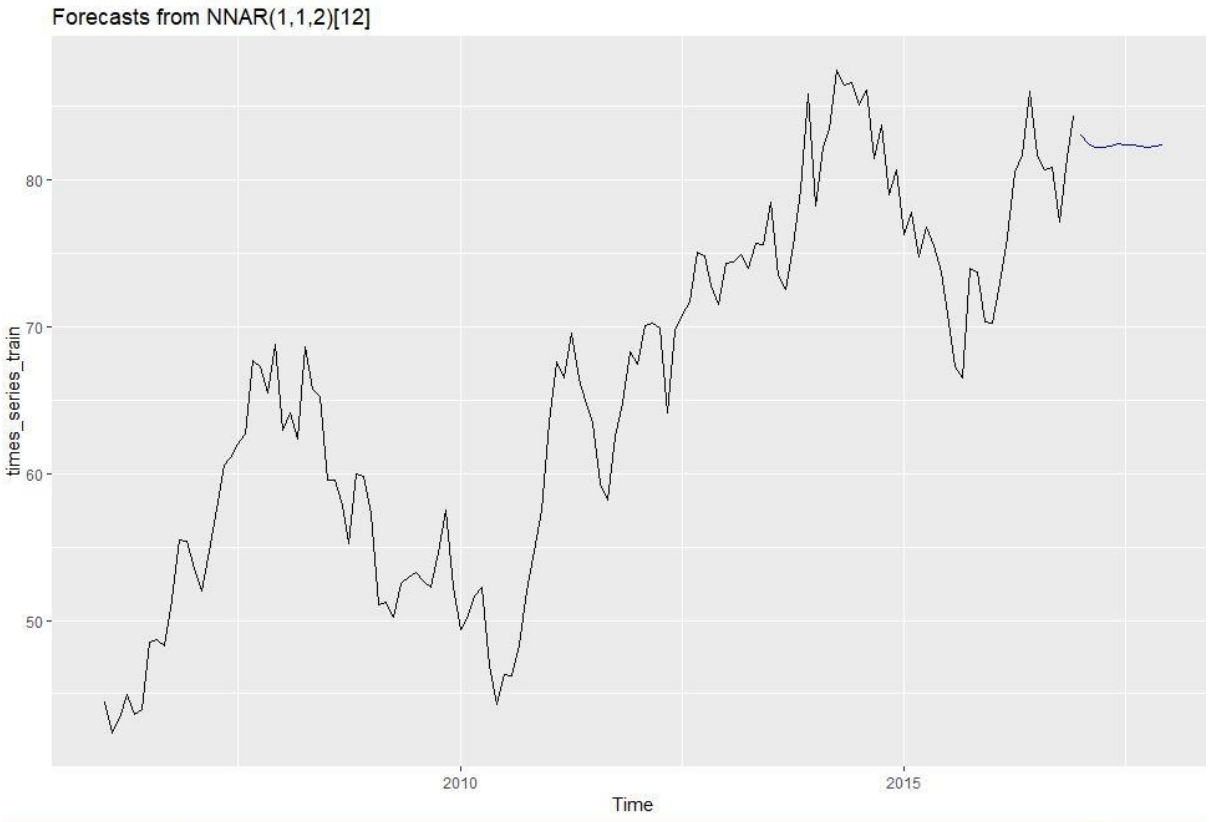
	ME	RMSE	MAE	MPE	MAPE
ARIMA	-6.571059	6.901295	6.571059	- 8.530068	8.530068
Neural Net	-4.631801	5.097959	4.631801	-6.034784	6.034784
Expo Smooth	-6.570775	6.901024	6.570775	-8.529703	8.529703
TBATS	-6.484252	6.818694	6.484252	-8.418359	8.418359

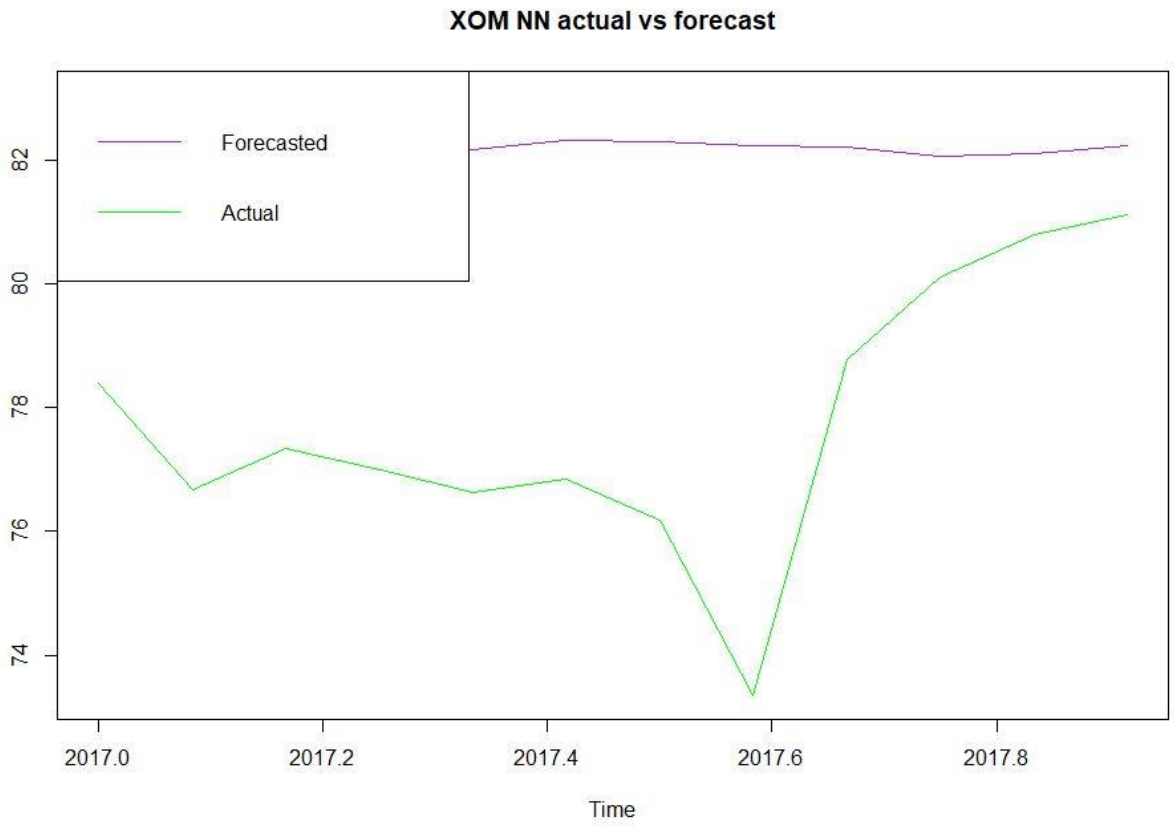
ARIMA forecast plots:


Forecasts from ARIMA(0,1,0)
100
80
60
40
times_series_train
2010
2015
Time
level
80
95

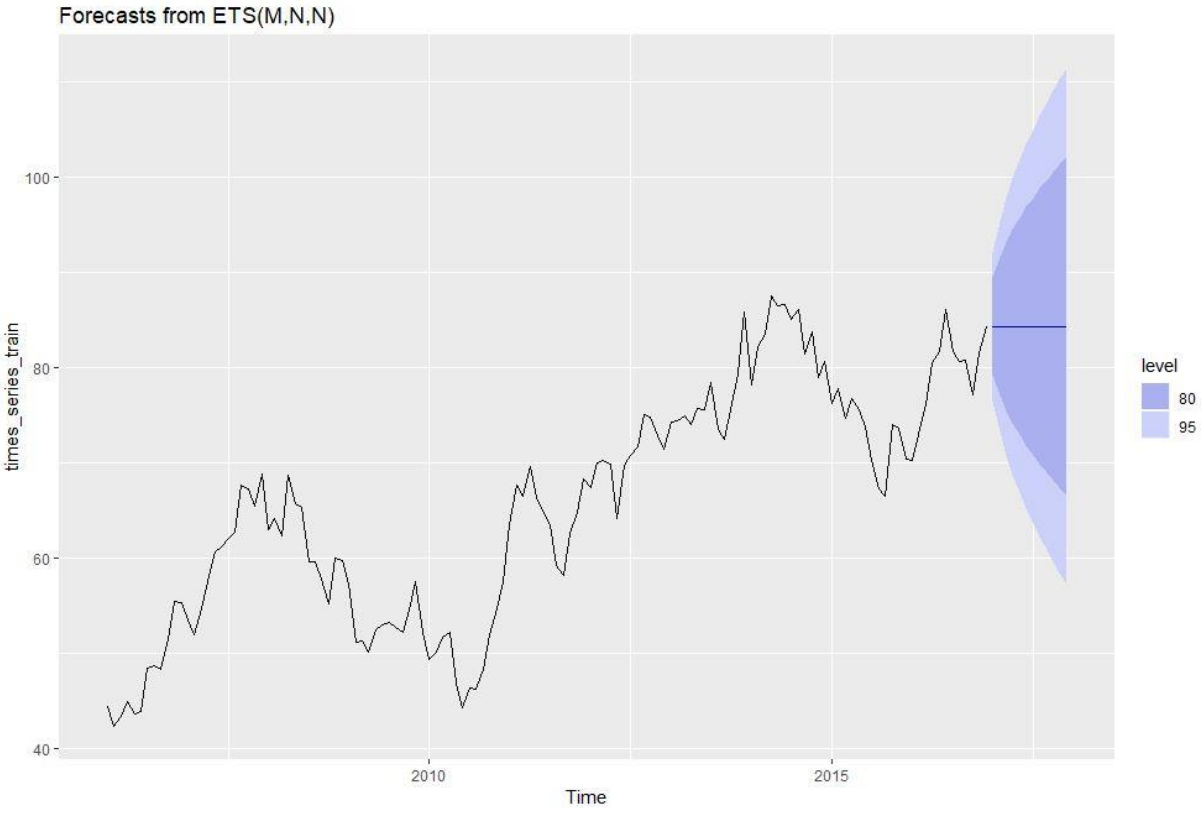

XOM ARIMA actual vs forecast
Forecasted
Actual
84
82
80
78
76
74
2017.0
2017.2
2017.4
2017.6
2017.8
Time

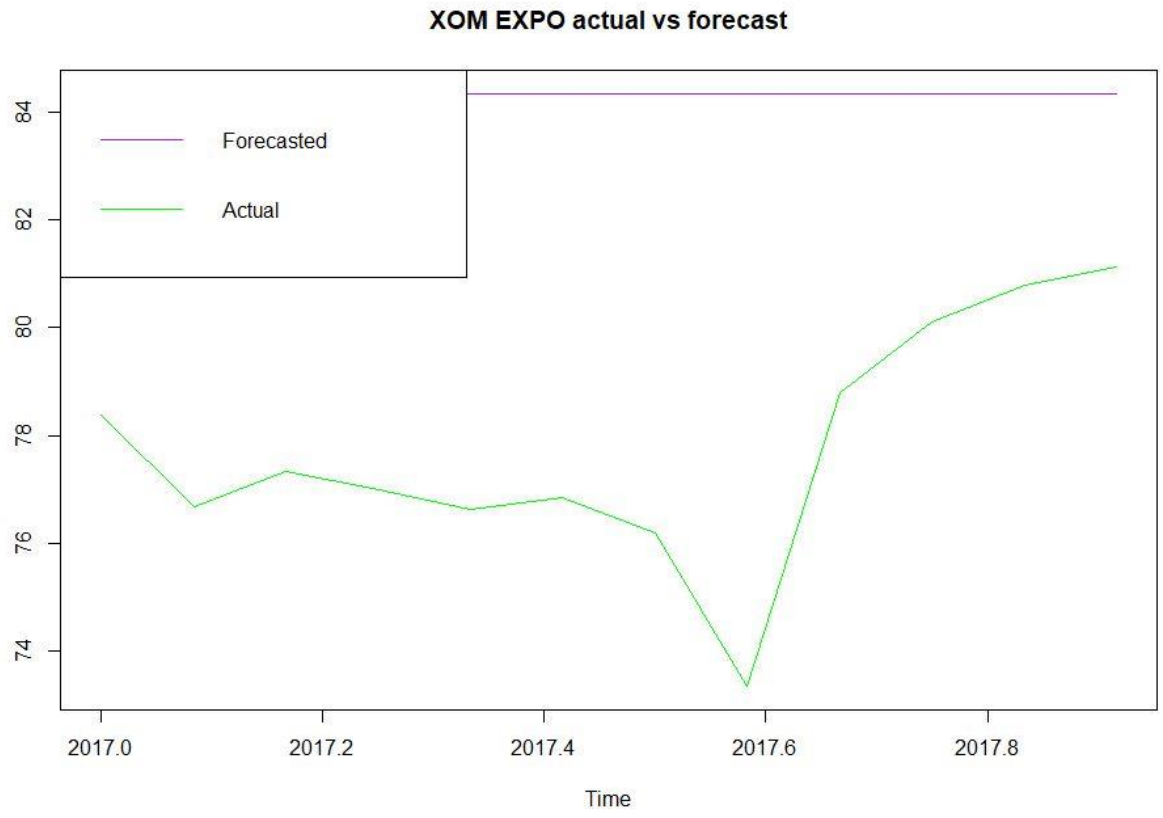
Neural Network:



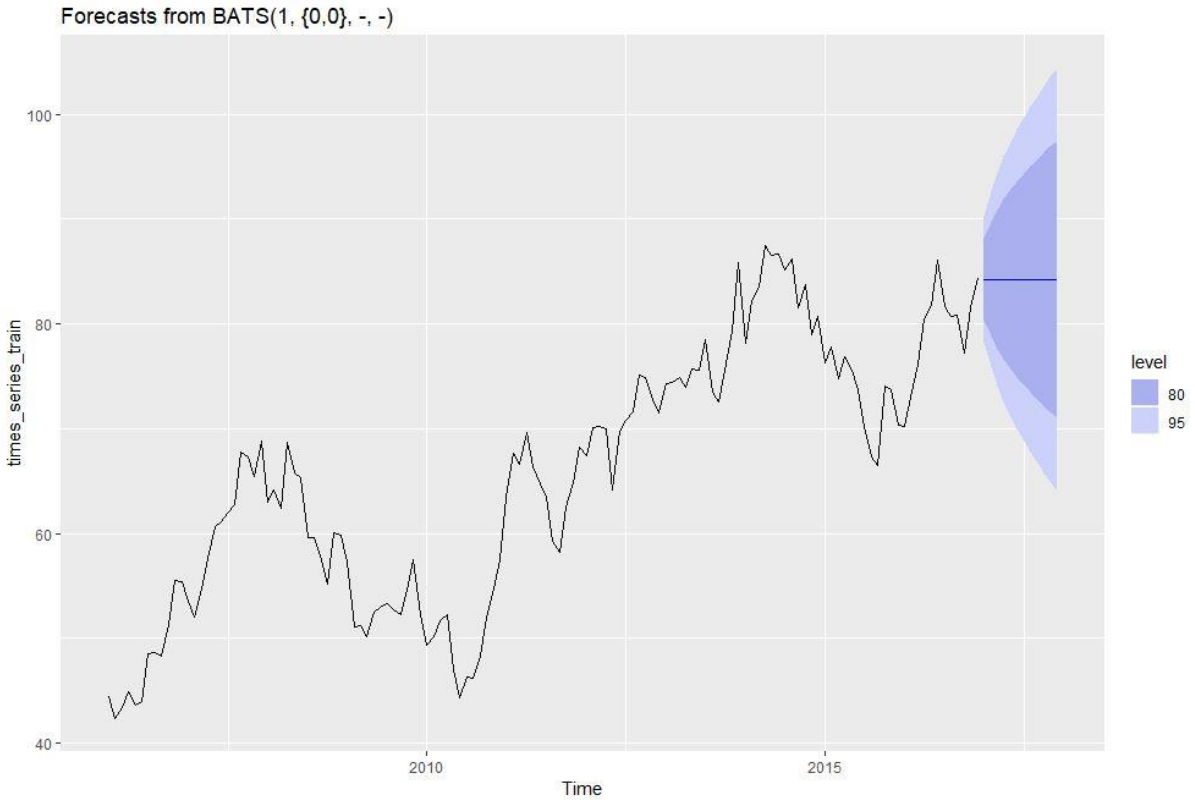


Exponential Smoothing:





TBATS:



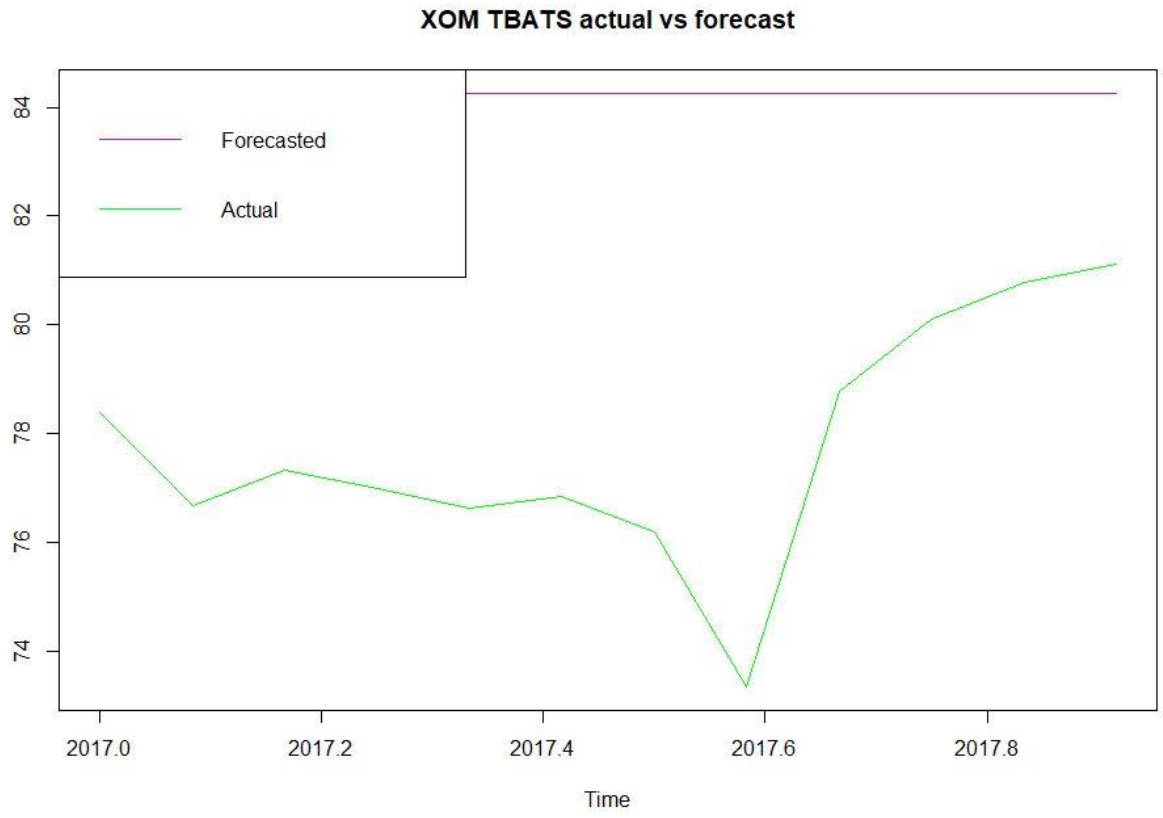


TABLEAU APPENDIX

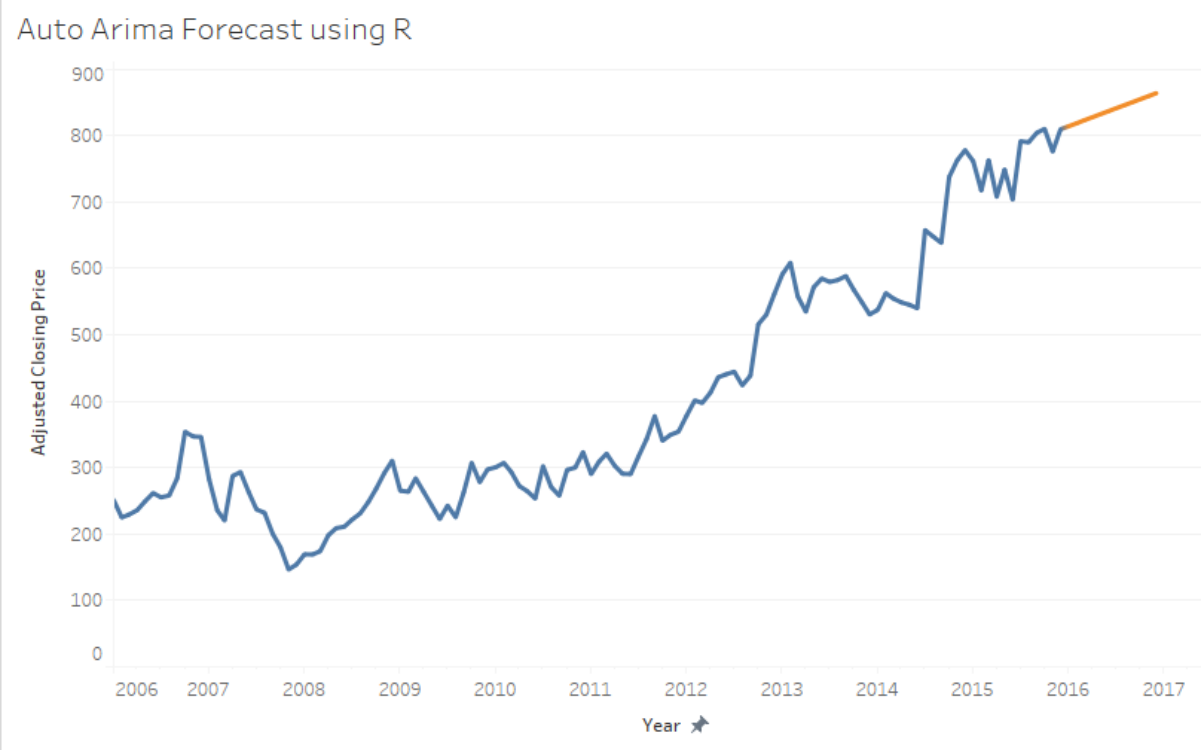
Please find included a Tableau packaged workbook file that includes the following:

- An ARIMA time series model applied to Google (daily) stock prices using the auto.arima function to predict a years worth , which uses the R package Rserve.
- As a comparision use of the Tableau exponential forecast algorithm.

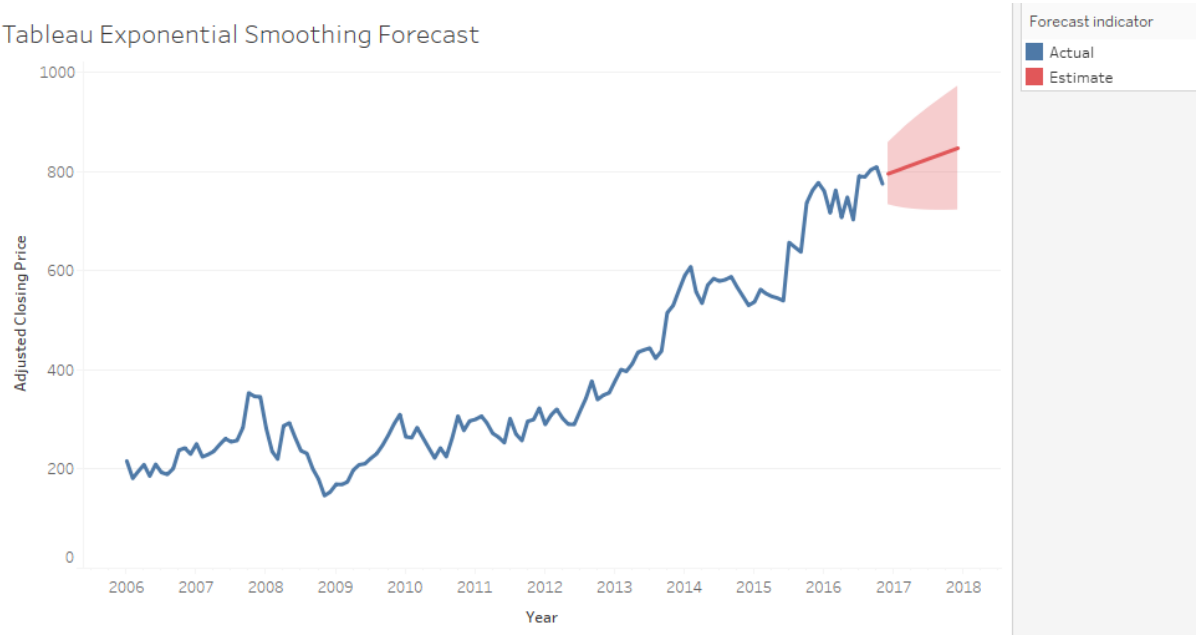
Note ensure you are running the Rserve package in the base R GUI and not RStudio. The following commands are as below in order to setup the connection between R and Tableau:

```
> library(Rserve)
> Rserve()
Starting Rserve...
 "C:\Users\robly\DOCUME~1\R\WIN-LI~1\3.5\Rserve\libs\x64\Rserve.exe"
>
```

The auto.arima function appears to predict a linear upward trend line. Note I was unable to produce confidence intervals in Tableau for this plot.



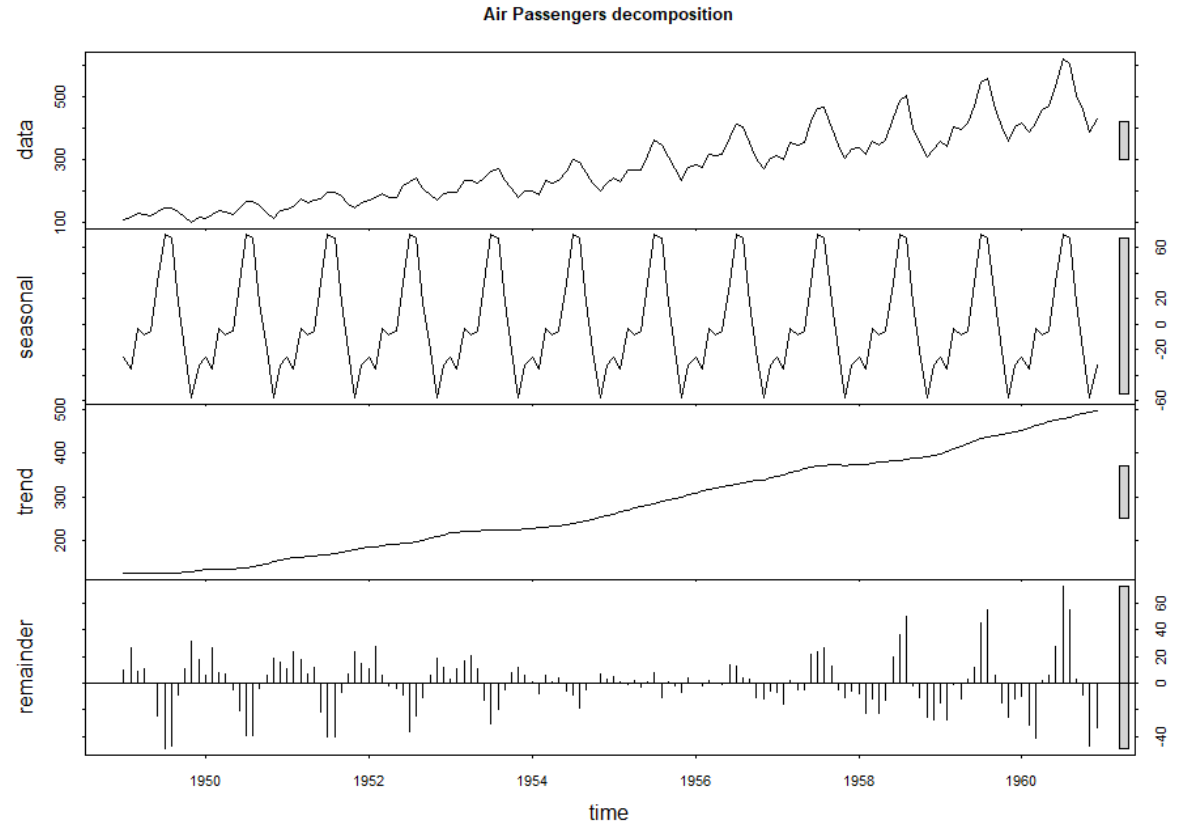
Below is the plot of Tableau's exponential smoothing forecast on the same Google dataset. Here a similar prediction to the auto.arima model above however, we can see the confidence intervals are reasonably narrow that is when compared to the confidence interval bands I was producing in R in the main section of this project.



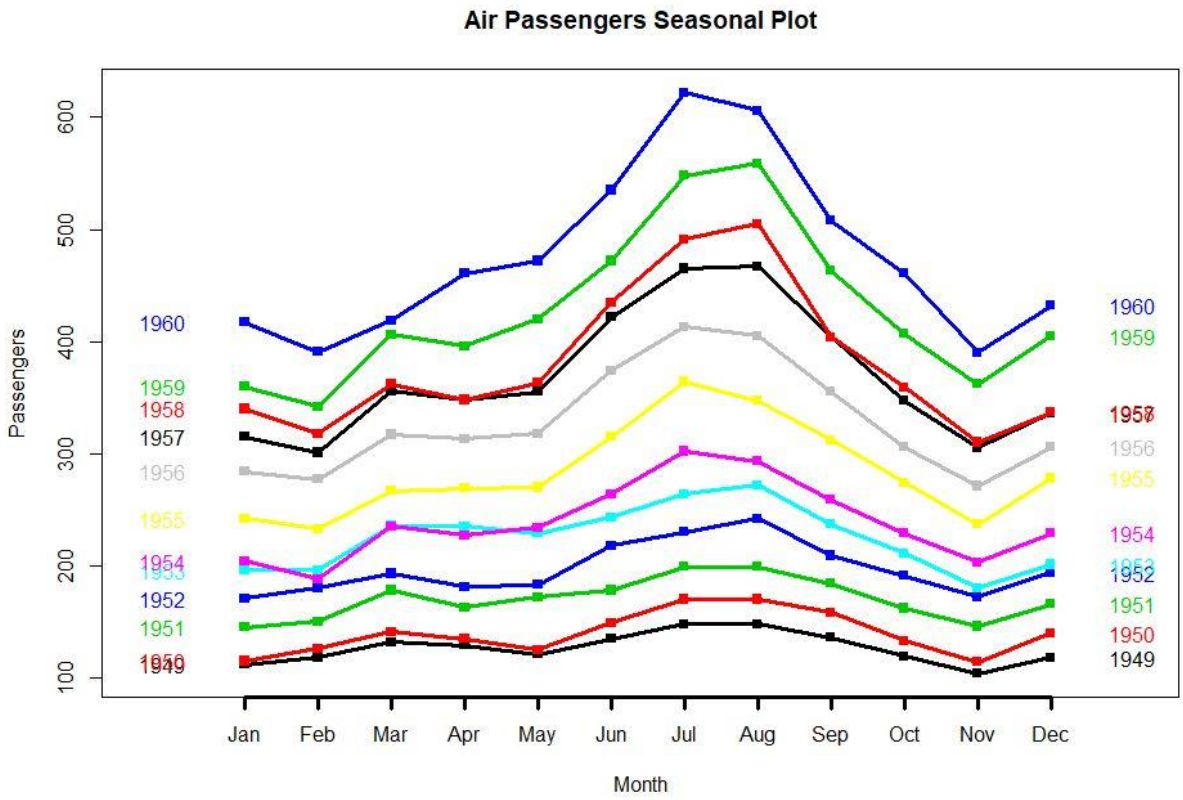
Air Passengers Appendix

I applied an ARIMA model on the classic well-known Air Passengers dataset which is available in R. This is a seasonal time series and I wanted to see how ARIMA would perform on it. Below are the charts and results.

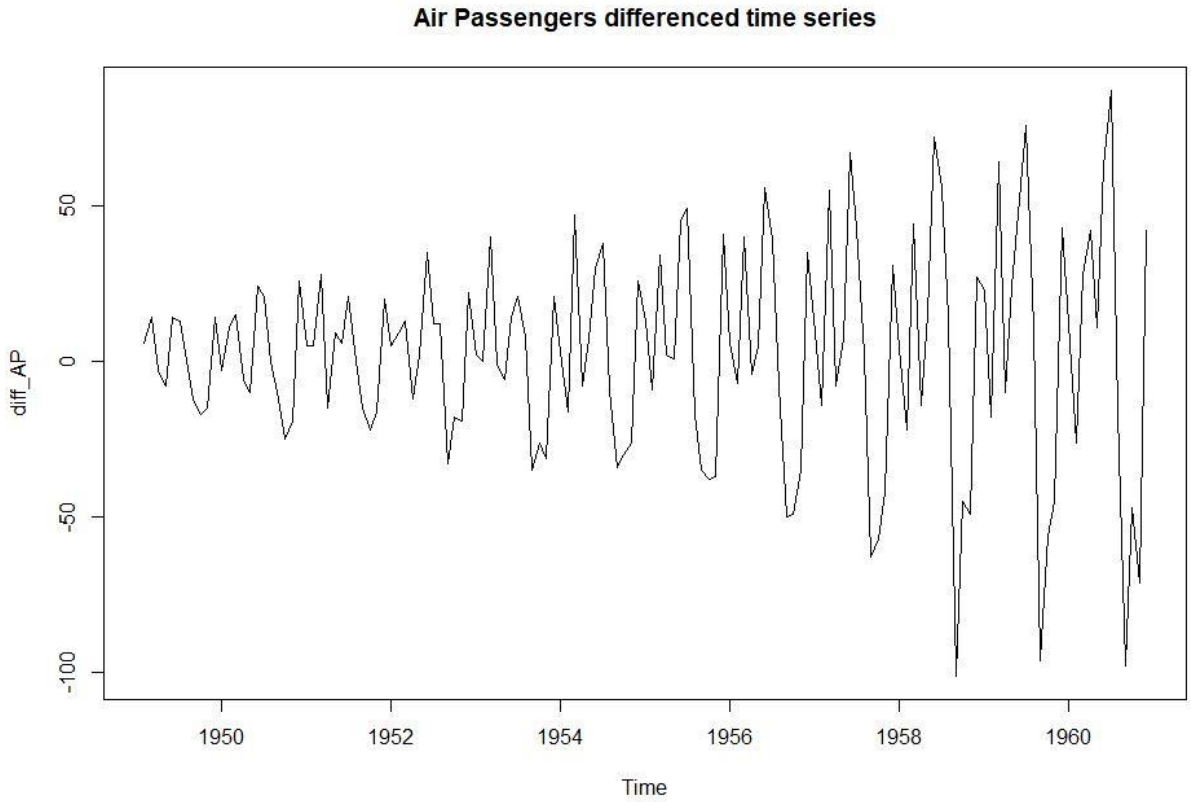
Seasonal, trend, and residuals plot:



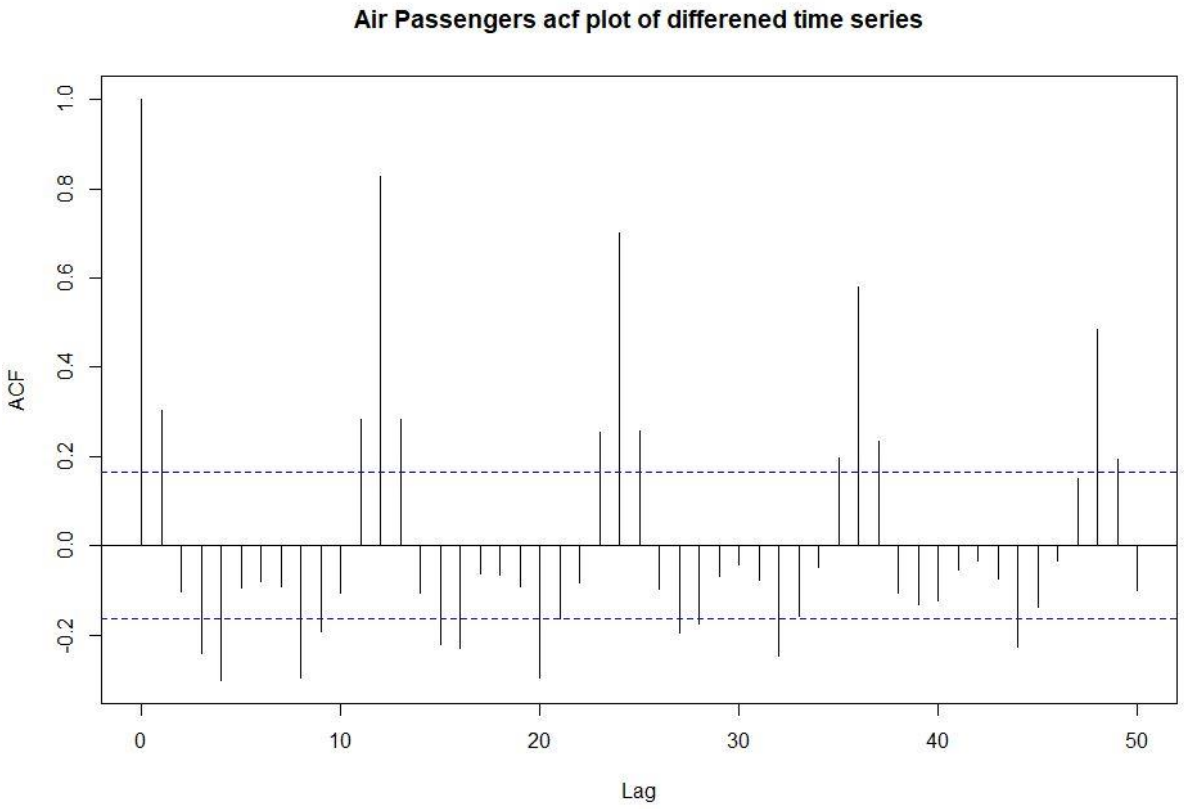
Seasonal inspect – clear monthly patterns each year:



Difference the time series to investigate further – mean approx. zero now



ACF plot of the differenced time series – clear high and low pattern here indicating seasonality

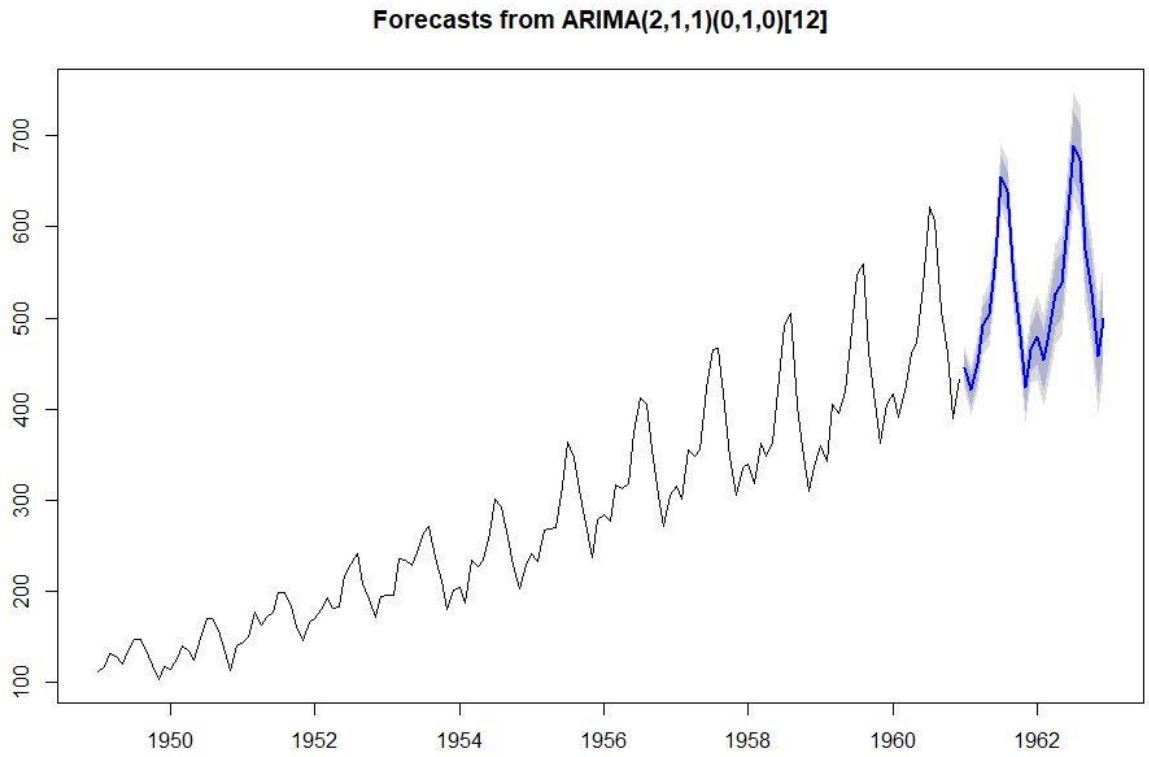


Summary of auto.arima model implementation

```
# Series: AP
# ARIMA(2,1,1)(0,1,0)[12]
#
# Coefficients:
#          ar1     ar2      ma1
#       0.5960  0.2143  -0.9819
# s.e.  0.0888  0.0880   0.0292
#
# sigma^2 estimated as 132.3:  log likelihood=-504.92
# AIC=1017.85   AICc=1018.17   BIC=1029.35
#
# Training set error measures:
#                    ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
# Training set 1.3423 10.84619 7.86754 0.420698 2.800458 0.245628 -0.00124847
```

ARIMA model (2,1,1)(0,1,0)[12] indicating AR(2), integrated once, MA (2). The seasonal element is captured (0,1,0) and [12] indicating the time series is monthly. All this information was automatically established by the auto.arima function.

Forecast 2 years ahead using the forecast function. The ARIMA model is excellent at capturing the seasonal trend into the future for this time series. The lighter coloured confidence intervals are in really good shape too and align very closely with the predictions.



GARCH APPENDIX

Heteroskedasticity is when the standard deviation or variance of a given variable is non-constant (ever present in the stock market). Conditional heteroskedasticity can help identify non-constant volatility when future volatility cannot be identified.

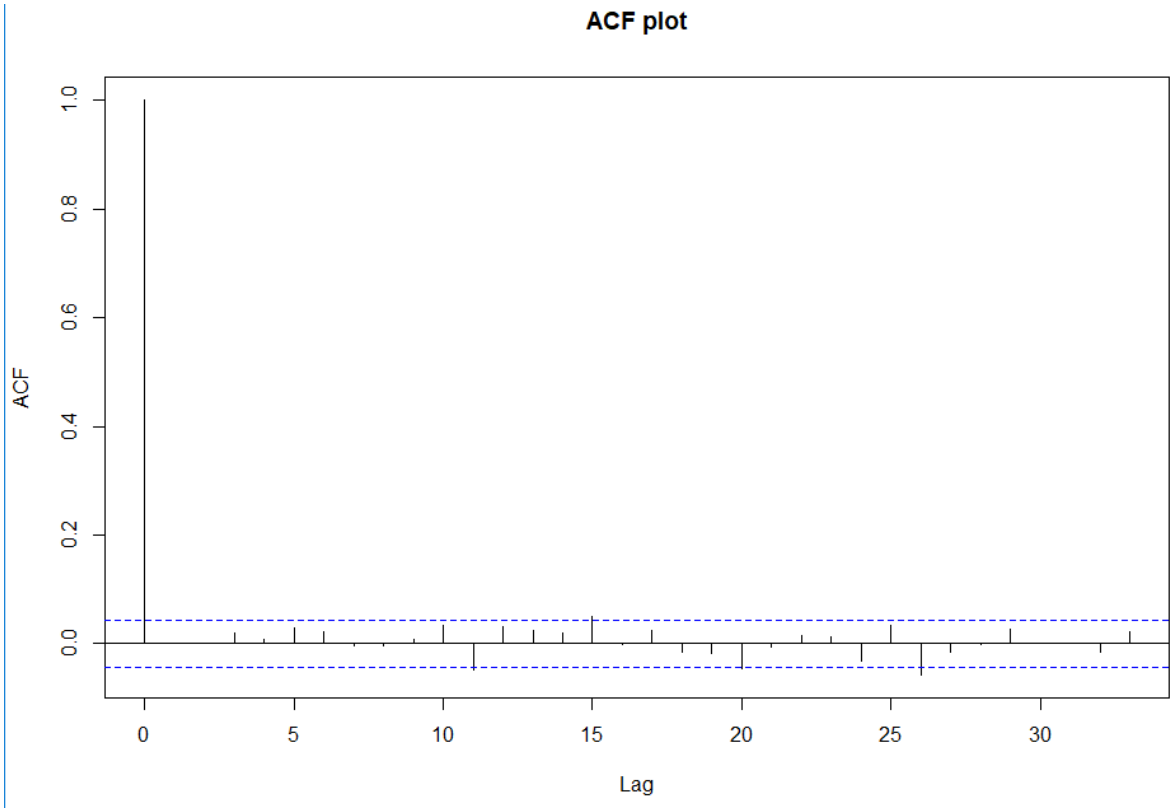
The **GARCH** model helps to model volatility clustering however it does not explain it. The variance of a residual error series follows an AR process of order p i.e. a random walk. An ARCH model uses AR(p) for modelling variance so for a GARCH model we combine AR(p) and MA(q) processes to model variance. The MA(q) model considers volatilities in the past. The GARCH model (p,q) formula:

$$e_t = \sigma_t w_t$$

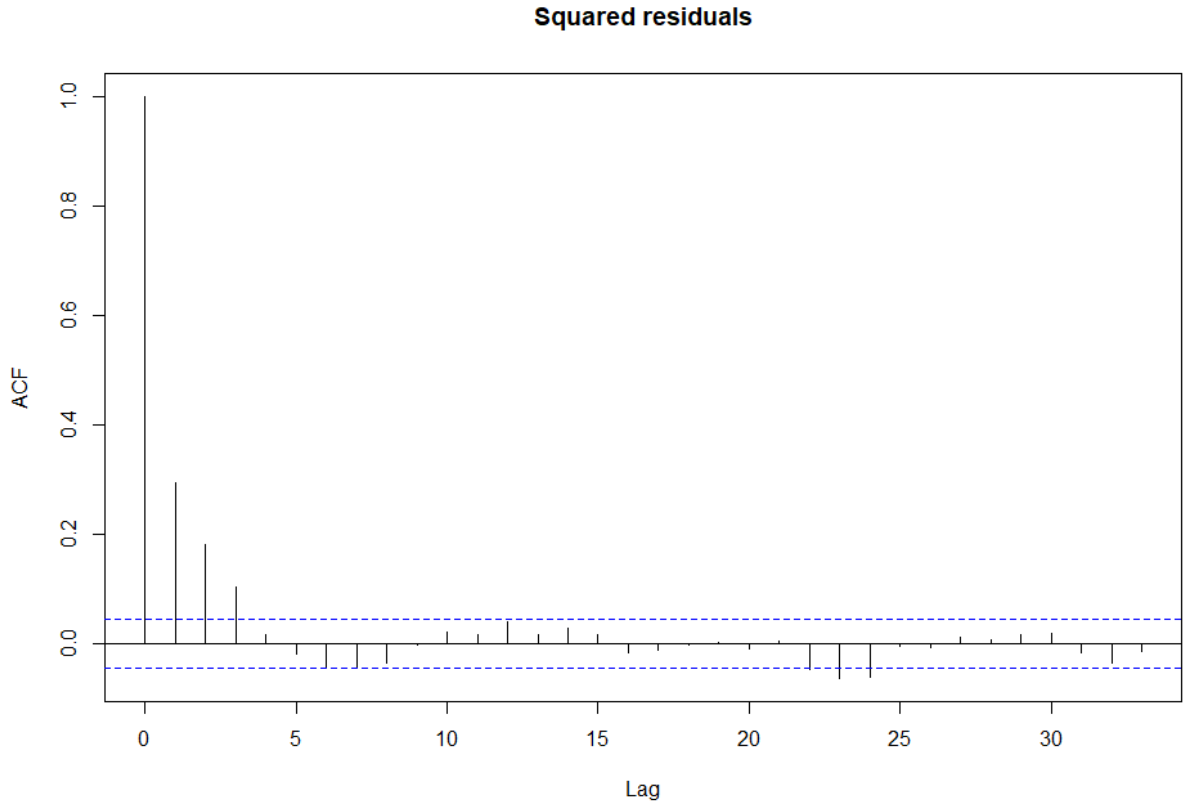
Note that e here is the residual error series which is the difference between the actual values and predicted values, so if there is no autocorrelation this indicates a good model. It is hard to detect Heteroskedasticity, so if we take a closer look at the autocorrelation plot it can be very similar to a stationary white noise process. Therefore, we have to analyse the square of the residual series – in order to decide whether we have to deal with volatility clustering or not.

Please find included the R file "GARCH". Here I simulate a GARCH model and use the GARCH function to try and explain the volatility present in the data. I examined the ACF plot and discover white noise however on plotting the ACF for the squared residuals there is slowly decaying autocorrelation meaning volatility is still present – the model doesn't seem to be able to explain all of the volatility. However, I was able to replicate the alpha and beta parameters of the model meaning the GARCH model is working quite well in explaining volatility.

ACF plot of residuals:



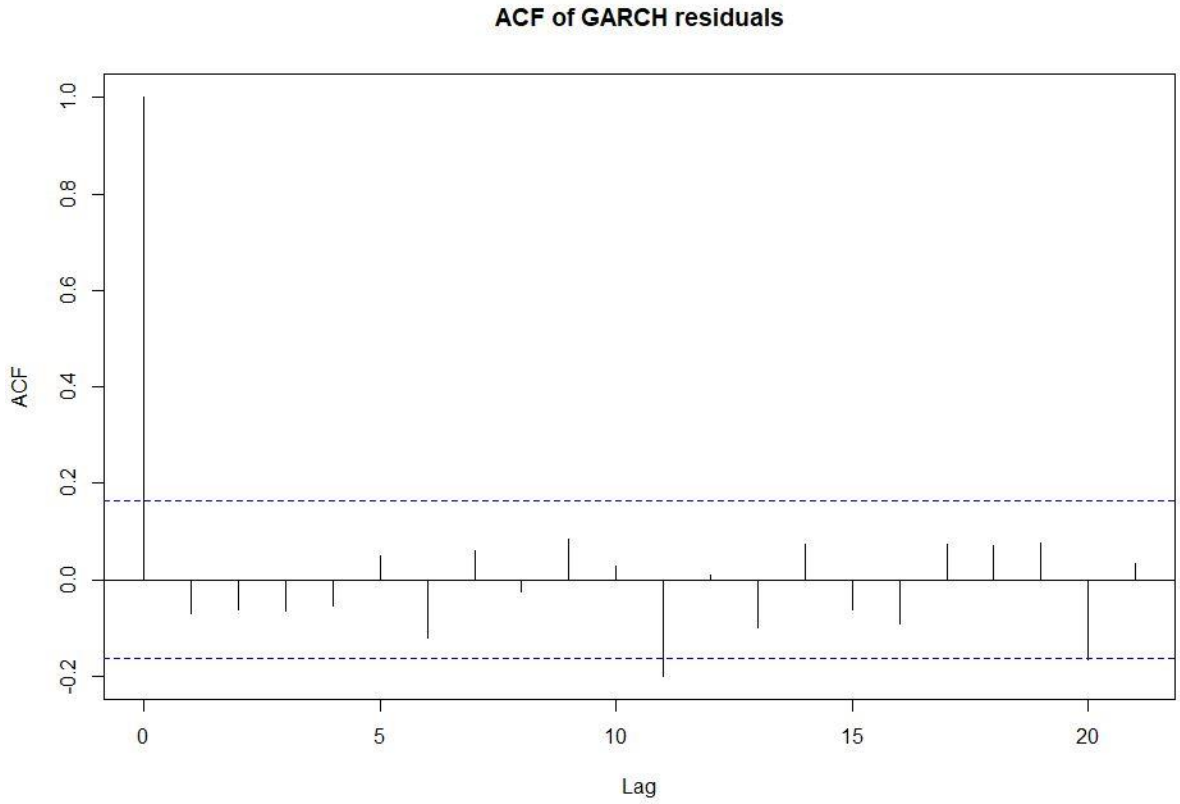
ACF plot of the squared residuals:



Please find included the R file "GARCH ARIMA GOOGL". Here I apply an ARIMA model to the 11 years' worth of Google stock price data followed by a GARCH model to try and explain volatility. Again, I examined the ACF plots of the GARCH residuals and discovered approx. white noise however

on plotting the squared residuals there appears to be random correlations still present. The GARCH model doesn't seem to be able to explain all of the volatility once again but does a pretty good job.

ACF of GARCH residuals:



ACF of GARCH squared residuals: