**Time series Analysis Project**

**For forecasting**

**The closing price**

**Of a stock**

**By**

**Yatin-Gupta**

**Email Id**

**yatin.gupta9814@gmail.com**

**Objectives**

The main objective of this project is to conduct the time series analysis and use the concepts studied by me in the subject CS-2 (Risk Modelling and Survival Analysis) of Actuarial Science. To try and apply the knowledge acquired in a more practical manner and learn some more while researching for the topic.

**Introduction**

Time Series is a sequence of well-defined data points measured at consistent time intervals over a period of time. Time series analysis is the use of statistical methods to analyze time series data and extract meaningful statistics and characteristics about the data.

We’ll dive into the implementation part of this project soon, but first it’s important to establish what we’re aiming to solve. Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

* Fundamental Analysis involves analyzing the company’s future profitability on the basis of its current business environment and financial performance.
* Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

As you might have guessed, our focus will be on the technical analysis part.

We will use R to perform time series analysis on dataset and all the R codes are highlighted in yellow.

**Packages required**

There are three packages required namely readxl, forecast, tseries.

The R code for loading these packages is given below:

library(“readxl”)

library(“forecast”)

library(“tseries”)

**Get Data**

Now we collect our data. I have collected monthly data for years (2006-2017) of 1 of 30 DJIA companies “AABA” (a total of 144 observations) from Kaggle. I have divided the data into two parts for training and then testing our model.

**Loading the Data in R**

ts\_training<-read\_xlsx("C:/Users/hp/Desktop/ts\_training.xlsx",sheet=1)

ts\_testing<-read\_xlsx("C:/Users/hp/Desktop/ts\_testing.xlsx",sheet=1)

**Getting started**

First our testing and training data has been converted to time series object in R.

Then our training data is summarized using head, end, start, frequency functions.

We start our analysis by plotting our training time series object to give us a visual basis to start our modeling. ACF and PACF is saved in ‘a’ and ‘p’ respectively.

**R code**

> ts\_training<-ts(ts\_training,start=c(2006,1),end=c(2016,12),frequency=12)

> ts\_testing<-ts(ts\_testing,start=c(2017,1),frequency=12)

> head(ts\_training)

Jan Feb Mar Apr May Jun

2006 38.11300 32.97579 31.21870 32.28316 31.51727 30.98182

> end(ts\_training)

[1] 2016 12

> start(ts\_training)

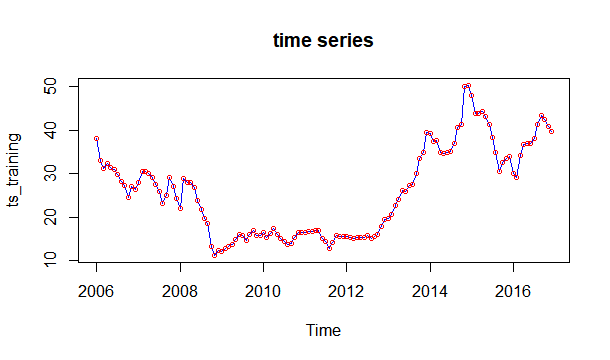
[1] 2006 1

> frequency(ts\_training)

[1] 12

> ts.plot(ts\_training,main="time series",col="blue")

> points(ts\_training,col="red",cex=0.7)



> a<-acf(ts\_training,lag.max=100,plot=FALSE)

> p<-pacf(ts\_training,lag.max=100,plot=FALSE)

**Testing stationarity**

A stochastic process is called stationary if the mean and variance are constant (i.e., their joint distribution does not change over time).

To test stationarity 3 methods are used.

**1st method:**

If the sample ACF is decreasing slowly and steadily then the data should be differenced.

**R code:**

> a<-acf(ts\_training,lag.max=100,plot=FALSE)

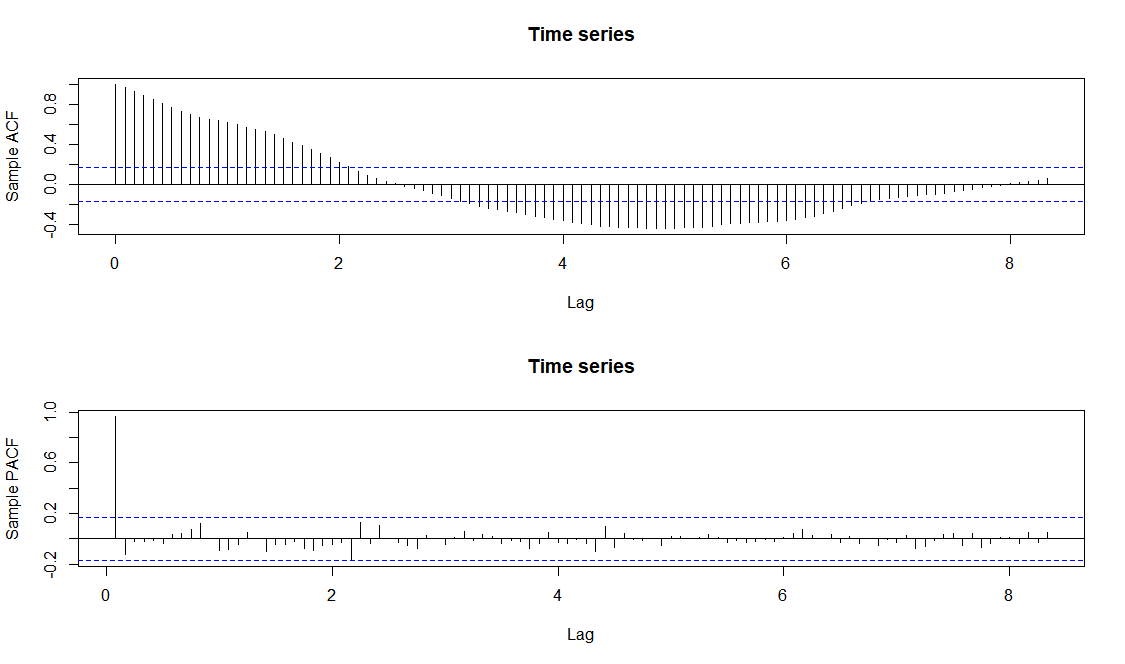
> p<-pacf(ts\_training,lag.max=100,plot=FALSE)

> par(mfrow=c(2,1))

> acf(ts\_training,main="Time series",lag.max=100,ylab="Sample ACF")

> pacf(ts\_training,main="Time series",lag.max=100,ylab="Sample PACF")

> par(mfrow=c(1,1))



The sample ACF does not quickly falls inside the confidence interval. This indicates that the time series needs to be differenced.

**2nd method:**

**PP test**

In [statistics](https://en.wikipedia.org/wiki/Statistics), the Phillips–Perron test (named after [Peter C. B. Phillips](https://en.wikipedia.org/wiki/Peter_C._B._Phillips) and [Pierre Perron](https://en.wikipedia.org/wiki/Pierre_Perron)) is a [unit root](https://en.wikipedia.org/wiki/Unit_root) test. That is, it is used in [time series](https://en.wikipedia.org/wiki/Time_series) analysis to test the [null hypothesis](https://en.wikipedia.org/wiki/Null_hypothesis) that a time series is [integrated of order](https://en.wikipedia.org/wiki/Order_of_integration) 1. It builds on the [Dickey–Fuller test](https://en.wikipedia.org/wiki/Dickey%E2%80%93Fuller_test) of the null hypothesis. {\displaystyle \rho =1}{\displaystyle \Delta y\_{t}=(\rho -1)y\_{t-1}+u\_{t}\,}{\displaystyle \Delta }{\displaystyle y\_{t}}{\displaystyle y\_{t-1}}Whilst the augmented Dickey–Fuller test addresses this issue by introducing lags {\displaystyle \Delta y\_{t}}as regressors in the test equation, the Phillips–Perron test makes a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) correction to the t-test statistic. The test is robust with respect to unspecified [autocorrelation](https://en.wikipedia.org/wiki/Autocorrelation) and [heteroscedasticity](https://en.wikipedia.org/wiki/Heteroscedasticity) in the disturbance process of the test equation.

H0: the time series has a unit root (and hence can be differenced)

H1: the time series does not need to be differenced

**R code**

> PP.test(ts\_training)

Phillips-Perron Unit Root Test

data: ts\_training

Dickey-Fuller = -2.3055, Truncation lag parameter = 4, p-value = 0.4495

The p‐value is greater than 0.05, so there is insufficient evidence at the 5% level to reject H0 and we conclude that the time series should be differenced at least once.

**3rd method:**

**ADF TEST**

Utilize the **Augmented Dickey-Fuller Test** for stationarity. The null hypothesis states that large p values indicate non-stationarity and smaller p values indicate stationarity. (We will be using 0.05 as our alpha value.)

**R code**

> adf.test(ts\_training)

Augmented Dickey-Fuller Test

data: ts\_training

Dickey-Fuller = -2.1154, Lag order = 5, p-value = 0.5285

alternative hypothesis: stationary

You can see our p value for the ADF test is relatively high. For that reason, we need to do some further visual inspection -but we know we will most likely have to difference our time series for stationarity.

**Differencing for achieving stationarity**

Now we will difference the series once to try and achieve stationary and then checking if it is achieved or requires differencing more times

**R code:**

> Xt<-diff(ts\_training,lag=1,differences=1)

> par(mfrow=c(2,2))

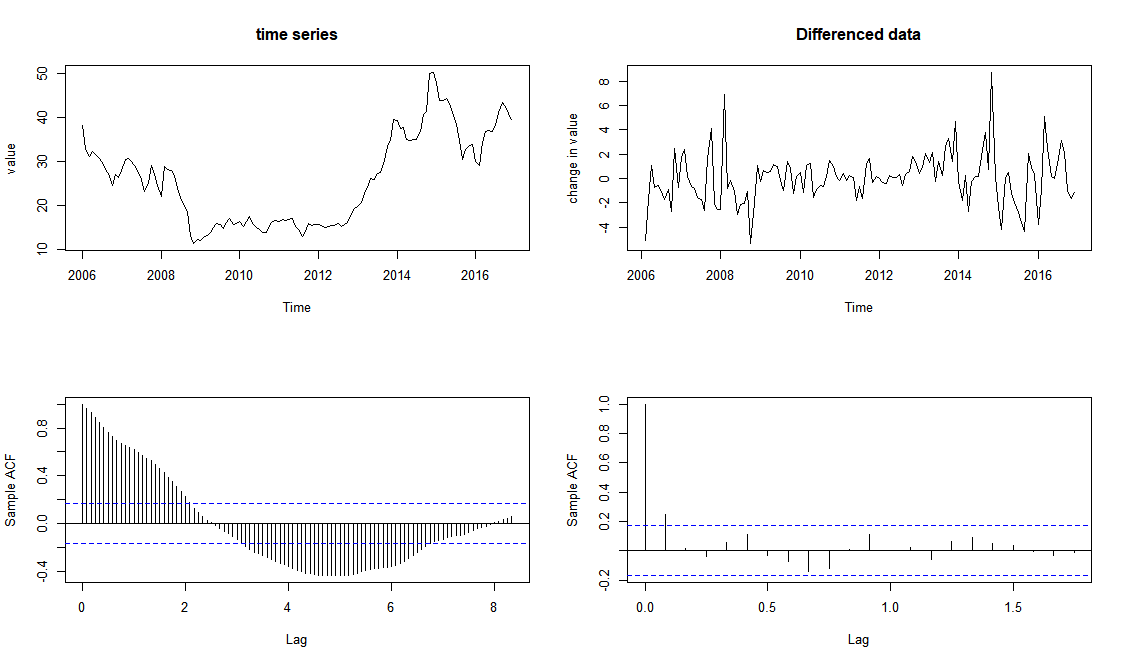
> ts.plot(ts\_training,main="time series",ylab="value")

> ts.plot(Xt,main="Differenced data",ylab="change in value")

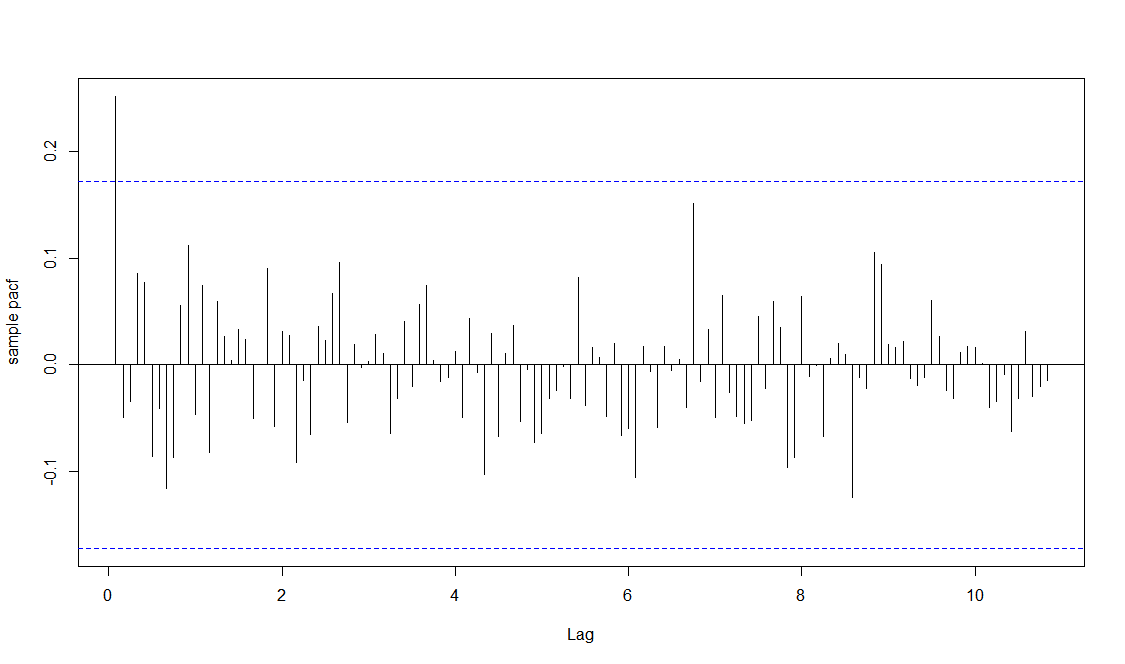
> acf(ts\_training,lag.max=100 ,main="",ylab= "Sample ACF")

> acf(Xt,main="",ylab= "Sample ACF")

> par(mfrow=c(1,1))



> pacf(Xt,lag.max=200,ylab="sample pacf",main="")



Looking at ACF and PACF for the differenced series, it is safe to say that stationarity has been achieved.

> PP.test(Xt)

Phillips-Perron Unit Root Test

data: Xt

Dickey-Fuller = -8.9717, Truncation lag parameter = 4, p-value = 0.01

The p‐value is less than 0.05, so we reject H0 and conclude that the time series does not need to be differenced a second time.

It may sometimes be unclear whether the data should be differenced a second time (or more). Recall that by differencing the data, the sample variance will normally decrease until stationarity is achieved. However, if we continue to difference the data after achieving stationarity, the sample variance will normally increase. In other words, we can set d to the value that minimizes the variance of the differenced data.

> var(ts\_training)

close

close 104.3696

> var(Xt)

close

close 4.260479

> d2t<-diff(ts\_training,lag=1,differences=2)

> var(d2t)

close

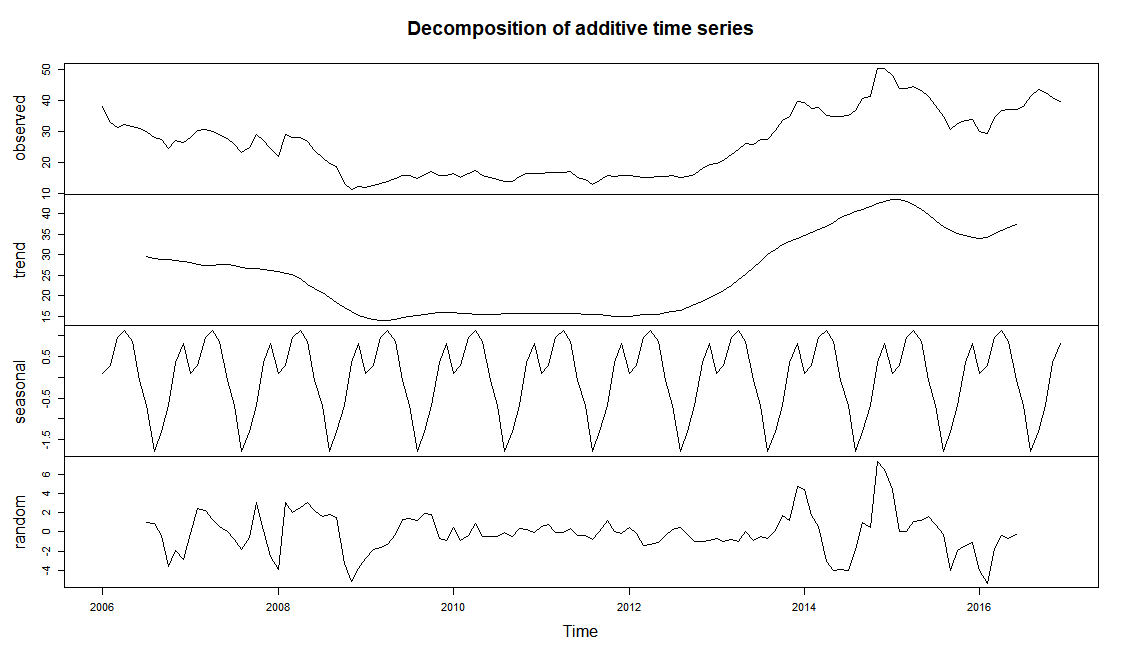
close 6.216185

This is higher than the variance of Xt, so we can conclude that the data should not be differenced a second time. Therefore we should fit an ARIMA(p,d,q) model with d = 1 .

**Testing and removing seasonality**

To further understand the anatomy of our data, we will break down our time series into its seasonal component, trend, and residuals using decompose function in R.

> plot(decompose(ts\_training,type="additive"))



> decomp<-decompose(ts\_training,type="additive")

> trend<-decomp$trend

> head(trend,7)

Jan Feb Mar Apr May Jun Jul

2006 NA NA NA NA NA NA 29.59917

> tail(trend,7)

Jun Jul Aug Sep Oct Nov Dec

2016 37.27249 NA NA NA NA NA NA

> seasonal<-decomp$seasonal

> random<-decomp$random

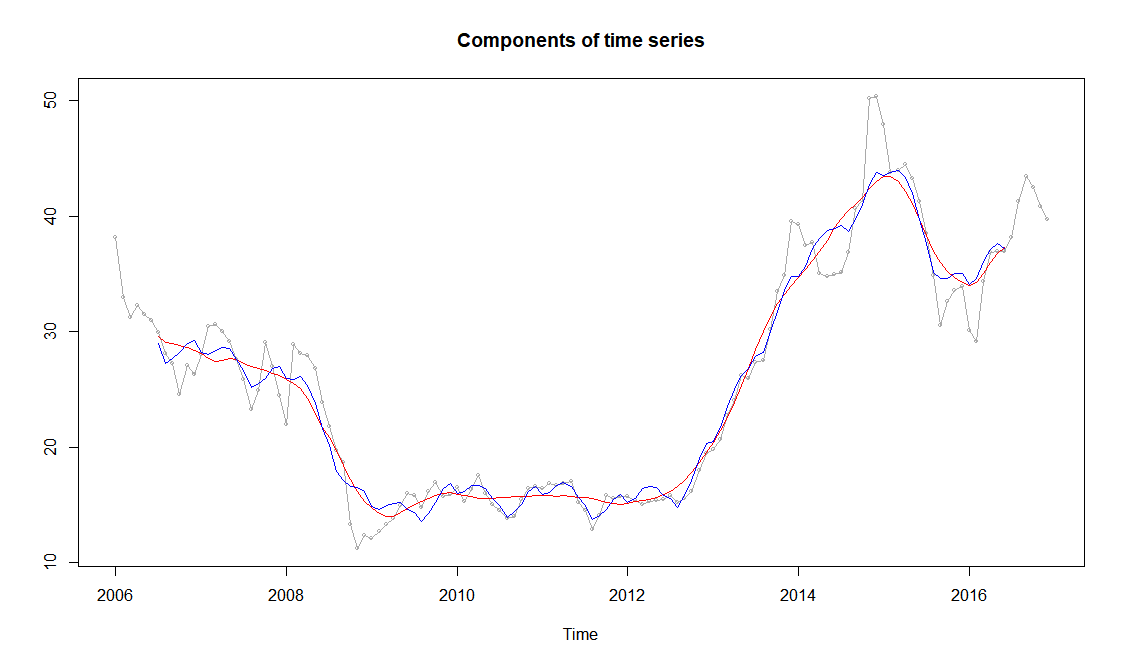
It is possible to plot our data, trend and seasonal components on the same graph.

> ts.plot(ts\_training,ylab="",main="Components of time series",col="dark grey")

> points(ts\_training,cex=0.5,col="dark grey")

> lines(trend,col="red")

> lines(seasonal+trend,col="blue")



**Fitting the model**

We will be fitting an **ARIMA** model. Autoregressive integrated moving average models are, in theory, the most general class of models for forecasting a time series which can be made to be “stationary” by differencing. An ARIMA model can be viewed as a “filter” that tries to separate the signal from the noise, and the signal is then extrapolated into the future to obtain forecasts.

We will be using **AIC** as the criteria for our model selection The Akaike information criterion (AIC) is an [estimator](https://en.wikipedia.org/wiki/Estimator) of the relative quality of [statistical models](https://en.wikipedia.org/wiki/Statistical_model) for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Thus, AIC provides a means for [model selection](https://en.wikipedia.org/wiki/Model_selection). In estimating the amount of information lost by a model, AIC deals with the trade-off between the [goodness of fit](https://en.wikipedia.org/wiki/Goodness_of_fit) of the model and the simplicity of the model. In other words, AIC deals with both the risk of [overfitting](https://en.wikipedia.org/wiki/Overfitting) and the risk of underfitting.

We will determine the order in which we are going to create our model.

**R code:**

> model=auto.arima(ts\_training)

> model$x.mean

NULL

> summary(model)

Series: ts\_training

ARIMA(1,1,0)

Coefficients:

ar1

0.2621

s.e. 0.0862

sigma^2 estimated as 3.978: log likelihood=-275.86

AIC=555.72 AICc=555.81 BIC=561.47

Training set error measures:

ME RMSE MAE MPE MAPE MASE ACF1

Training set 0.008041713 1.979369 1.394957 -0.1583187 5.486733 0.2228921 0.001658156

> model$aic

[1] 555.7167

auto.arima function selects the model with lowest aic.

**Testing the fit**

**1st method:**

Our next step is to run residual diagnostics to ensure our residuals are white noise under our initial assumptions.

**R code:**

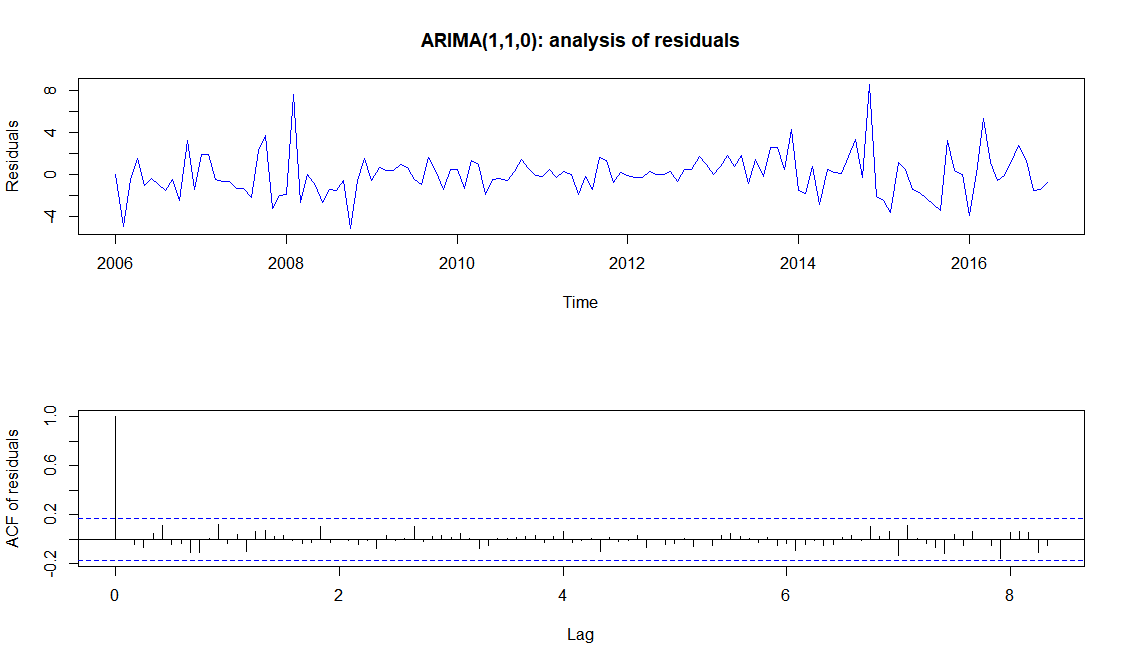
> e<-model$residuals

> par(mfrow=c(2,1))

> ts.plot(e,main="ARIMA(1,1,0): analysis of residuals", ylab="Residuals",col="blue")

> acf(e,main="",lag.max=100,ylab="ACF of residuals")

> par(mfrow=c(1,1))



The mean and the variance of the residuals are broadly constant over time. ACFs of the residuals are all small, with no significant pattern, so the residuals appear to be independent. Hence the residuals appear to be white noise, and we can conclude that the model is a good fit.

**2nd method:**

**Ljung Box Test**

The test determines whether or not errors are [iid](https://www.statisticshowto.datasciencecentral.com/iid-statistics/) (i.e. white noise) or whether there is something more behind them; whether or not the [autocorrelations](https://www.statisticshowto.datasciencecentral.com/serial-correlation-autocorrelation/)for the errors or residuals are non zero. Essentially, it is a test of *lack*of fit: if the autocorrelations of the [residuals](https://www.statisticshowto.datasciencecentral.com/residual/)are very small, we say that the model doesn’t show ‘significant lack of fit’.

H0: the residuals are independent

**R code:**

> Box.test(e,lag=5,type="Ljung",fitdf=2)

Box-Ljung test

data: e

X-squared = 2.966, df = 3, p-value = 0.3969

Since the p‐value is greater than 5%, there is insufficient evidence to reject H0 and we conclude that the residuals seems to be independent and model is a good fit.

**Forecasting**

Since we believe we've found the appropriate model, let's begin [forecasting](https://www.datascience.com/blog/how-to-forecast-with-limited-historical-data/)!

**R code:**

> forecast\_values=predict(model,n.ahead=12)

> forecast\_values

$pred

Jan Feb Mar Apr May Jun Jul Aug

2017 39.36540 39.28507 39.26401 39.25850 39.25705 39.25667 39.25657 39.25655

Sep Oct Nov Dec

2017 39.25654 39.25654 39.25654 39.25654

$se

Jan Feb Mar Apr May Jun Jul Aug

2017 1.994537 3.211666 4.166521 4.959525 5.646644 6.259841 6.818367 7.334546

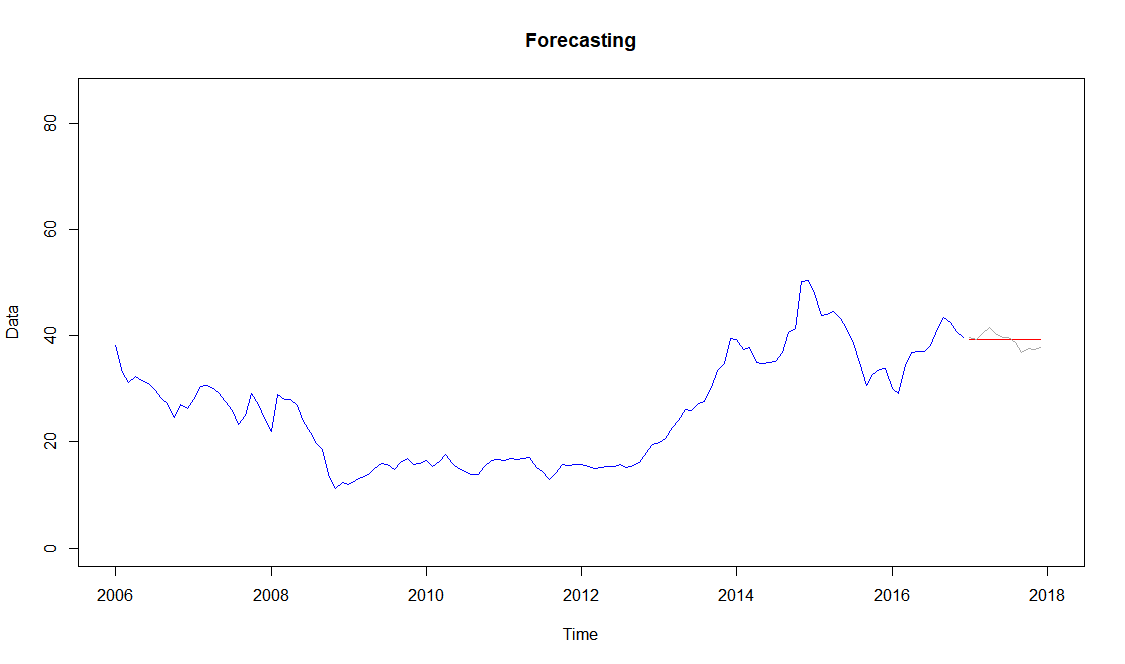
Sep Oct Nov Dec

2017 7.816728 8.270851 8.701306 9.111448

> ts.plot(ts\_training,col="blue",xlim=c(2006,2018),ylim=c(0,85),main="Forecasting",ylab="Data")

> lines(forecast\_values$pred,col="red")

> lines(ts\_testing,col="dark grey")



**Conclusions**

We can see that the forecasted values are near the actual closing price of stock for the year 2017. Thus we can say that we have created a good model for forecasting the closing stock prices for the stock “AABA”. Finally we can conclude that even though I have not used a more complex model which could better predict the stock prices, I have created a good model for prediction using the knowledge that I have. It is also well known that stock prices are highly volatile and depends on various other reasons and thus is extremely difficult to predict accurately

**References**

CS-2 course material

Wikipedia, Google searches for definitions and explanations