Time Series Assignment 1

Arindam Patra

03/11/2021

Name: Arindam Patra, Roll: MDS202008

Importing libraries

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, tidy = TRUE)
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.0.5
```

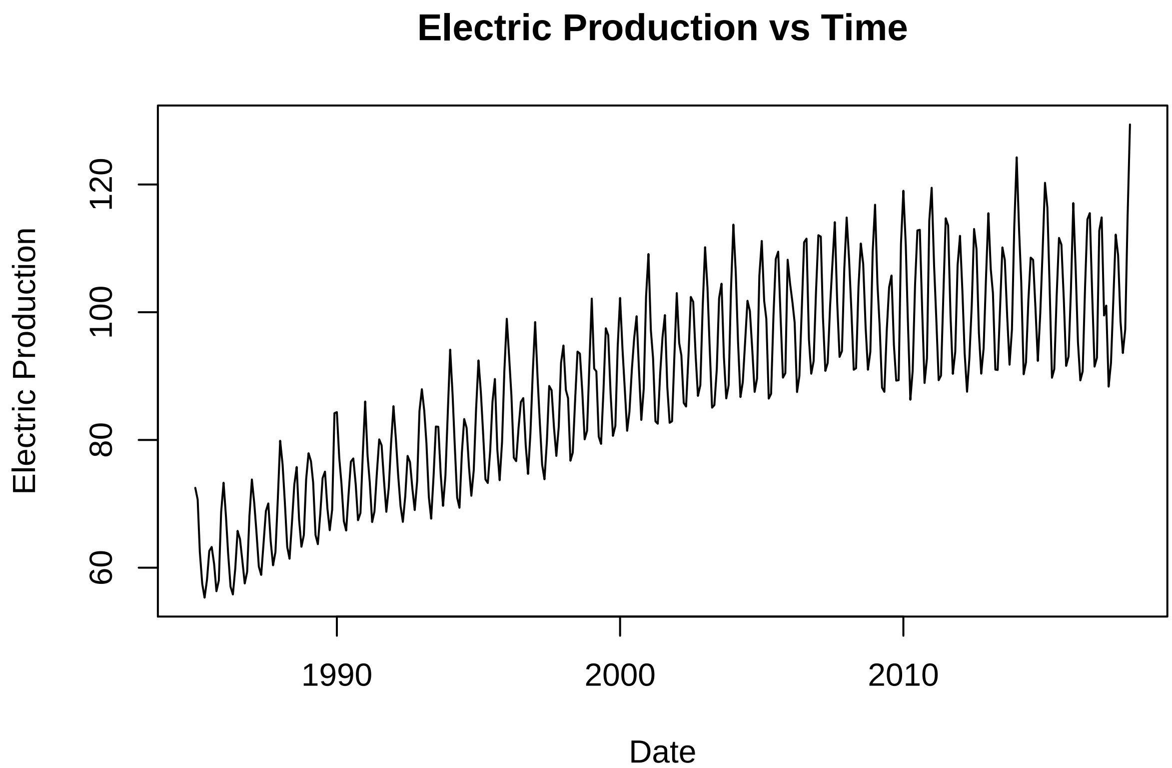
```
library(moments)
```

Importing Data

Monthly Electric Production data from 1985-2017

```
df <- read.csv("time_series/Electric_Production.csv",row.names="DATE")
data = ts(df)
Time <- as.Date(rownames(df),'%m/%d/%Y')

plot(Time ,data, xlab = 'Date',ylab = 'Electric Production',
     main = 'Electric Production vs Time', type = 'l')
```



Descriptive Statistics

```
m = mean(data)
s = sd(data)
print(paste("Minimum = ", min(data)))
```

```
## [1] "Minimum =  55.3151"
```

```
print(paste("Maximum = ", max(data)))
```

```
## [1] "Maximum =  129.4048"
```

```
print(paste("Range = ", max(data) - min(data)))
```

```
## [1] "Range =  74.0897"
```

```
print(paste("Mean = ", m))
```

```
## [1] "Mean =  88.8472176322418"
```

```
print(paste("Standard deviation = ", s))
```

```
## [1] "Standard deviation =  15.3878336647309"
```

```
print(paste("1st Quartile = ", quantile(data, 0.25)))
```

```
## [1] "1st Quartile =  77.1052"
```

```r
print(paste("2nd Quartile (Median) = ", quantile(data, 0.5)))
```

```
## [1] "2nd Quartile (Median) =  89.7795"
```

```r
print(paste("3rd Quartile = ", quantile(data, 0.75)))
```

```
## [1] "3rd Quartile =  100.5244"
```

```r
print(paste("Inter quartile range = ", IQR(data)))
```

```
## [1] "Inter quartile range =  23.4192"
```

```r
print(paste("Skewness = ", skewness(data)))
```

```
## [1] "Skewness =  -0.0728191541455781"
```

```r
print(paste("Kurtosis = ", kurtosis(data)))
```

```
## [1] "Kurtosis =  2.2994360495705"
```

```r
x = round(((length(data[data > m - (3 * s) & data < m + (3 * s)]) * 100)/length(data)),
          3)
print(paste("Percentage of observations in 3-sigma deviation of the mean = ", x, "%"))
```

```
## [1] "Percentage of observations in 3-sigma deviation of the mean =  100 %"
```

Since all the data lies between 3 SD, so the data doesn't have any outlier.

Stationarity Test

Let's test the stationarity of this time series. We will test it using 3 tests - ADF, KPSS and PP Test.

```r
adf = adf.test(data)
adf
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -5.139, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
kpss = kpss.test(data)
kpss
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  data
## KPSS Level = 6.3058, Truncation lag parameter = 5, p-value = 0.01
```

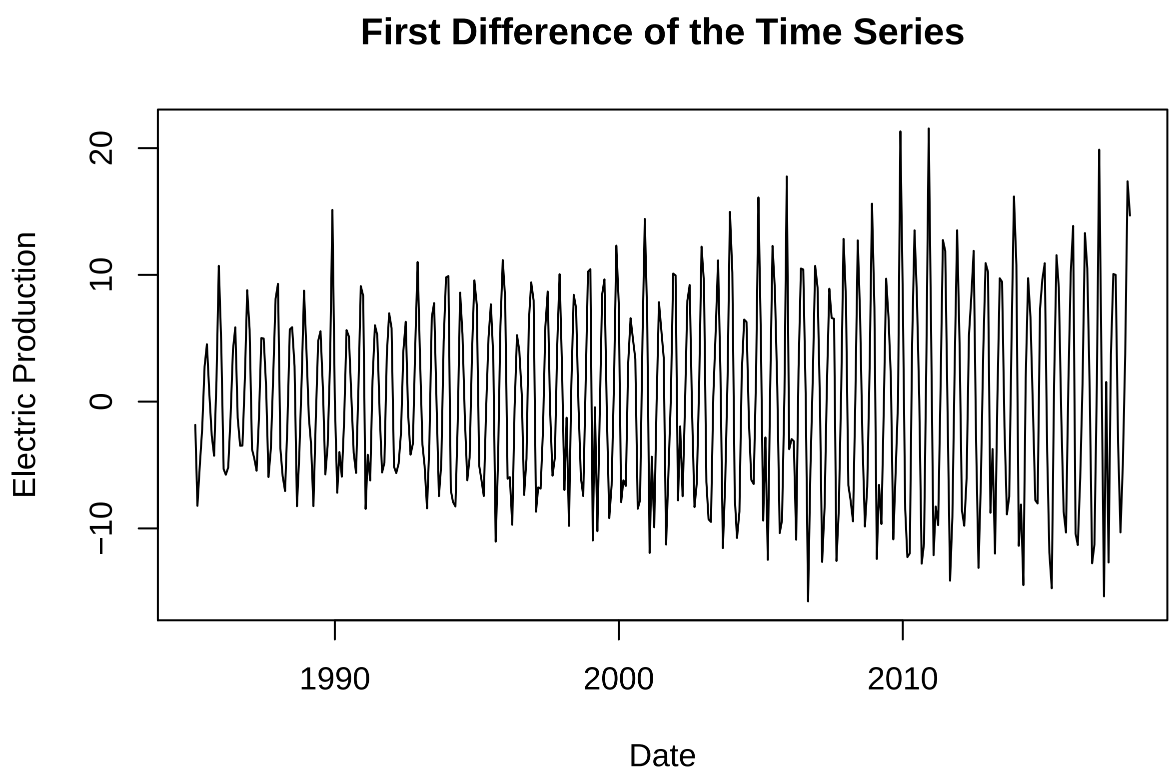
```r
pp = PP.test(data)
pp
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  data
## Dickey-Fuller = -8.2573, Truncation lag parameter = 5, p-value = 0.01
```

Here ADF and PP test are agreed for stationarity of this data but KPSS is saying non-stationary. So let's do the first difference of the data.

First Difference

```r
first_diff <- diff(data)
plot(Time[-1],first_diff, xlab = 'Date',ylab = 'Electric Production',
     main = 'First Difference of the Time Series',type = 'l')
```




```r
adf = adf.test(first_diff)
adf
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  first_diff
## Dickey-Fuller = -9.6447, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
kpss = kpss.test(first_diff)
kpss
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  first_diff
## KPSS Level = 0.078012, Truncation lag parameter = 5, p-value = 0.1
```

```
pp = PP.test(first_diff)
pp
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  first_diff
## Dickey-Fuller = -15.573, Truncation lag parameter = 5, p-value = 0.01
```

Ok! Here all of them agrees with the stationarity of this time series data. Therefore we can fit this data to our model.

Train test split

Keeping last 20 observations as test data and rest of them as train data.

```
test_size <- 20
n <- length(first_diff)
train_size <- n - test_size

train_data <- first_diff[c(1 : train_size)]
test_data <- first_diff[c((train_size+1) : n)]
```

```
### Total data points
length(first_diff)
```

```
## [1] 396
```

```
### Training Data Size
length(train_data)
```

```
## [1] 376
```

```
### Test Data Size
length(test_data)
```

```
## [1] 20
```

Modelling

At first we will build two optimized model. 1 - Using AIC, 2 - Using BIC

Using AIC

```
aic_model <- auto.arima(train_data, trace= TRUE,d= 0, max.p = 10, max.q = 10,
                        ic ="aic", approximation = FALSE)
```

```
##
##  ARIMA(2,0,2) with non-zero mean : 2082.145
##  ARIMA(0,0,0) with non-zero mean : 2593.348
##  ARIMA(1,0,0) with non-zero mean : 2536.677
##  ARIMA(0,0,1) with non-zero mean : 2470.228
##  ARIMA(0,0,0) with zero mean     : 2591.363
##  ARIMA(1,0,2) with non-zero mean : 2345.833
##  ARIMA(2,0,1) with non-zero mean : 2089.945
##  ARIMA(3,0,2) with non-zero mean : 2076.4
```

```
##  ARIMA(3,0,1) with non-zero mean : 2076.408
##  ARIMA(4,0,2) with non-zero mean : 2064.957
##  ARIMA(4,0,1) with non-zero mean : 2071.16
##  ARIMA(5,0,2) with non-zero mean : 2049.122
##  ARIMA(5,0,1) with non-zero mean : 2072.437
##  ARIMA(6,0,2) with non-zero mean : Inf
##  ARIMA(5,0,3) with non-zero mean : Inf
##  ARIMA(4,0,3) with non-zero mean : Inf
##  ARIMA(6,0,1) with non-zero mean : 2072.464
##  ARIMA(6,0,3) with non-zero mean : Inf
##  ARIMA(5,0,2) with zero mean     : 2060.629
##
##  Best model: ARIMA(5,0,2) with non-zero mean
```

```
aic_model
```

```
## Series: train_data
## ARIMA(5,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ar4     ar5      ma1      ma2    mean
##       0.6928  -0.3857  -0.2180  -0.0489  0.2877  -0.8282  -0.0991  0.1095
## s.e.  0.1446   0.1248   0.0998   0.0734  0.0573   0.1451   0.1311  0.0208
##
## sigma^2 estimated as 13.12:  log likelihood=-1015.56
## AIC=2049.12   AICc=2049.61   BIC=2084.49
```

For AIC, Best model is ARMA(5,2).

Using BIC

```
bic_model <- auto.arima(train_data, trace= TRUE,d=0, max.p = 10, max.q = 10,
                        ic ="bic", approximation = FALSE)
```

```
##
##  ARIMA(2,0,2) with non-zero mean : 2105.723
##  ARIMA(0,0,0) with non-zero mean : 2601.207
##  ARIMA(1,0,0) with non-zero mean : 2548.466
##  ARIMA(0,0,1) with non-zero mean : 2482.016
##  ARIMA(0,0,0) with zero mean     : 2595.293
##  ARIMA(1,0,2) with non-zero mean : 2365.481
##  ARIMA(2,0,1) with non-zero mean : 2109.593
##  ARIMA(3,0,2) with non-zero mean : 2103.907
##  ARIMA(3,0,1) with non-zero mean : 2099.985
##  ARIMA(3,0,0) with non-zero mean : 2166.644
##  ARIMA(4,0,1) with non-zero mean : 2098.667
##  ARIMA(4,0,0) with non-zero mean : 2093.737
##  ARIMA(5,0,0) with non-zero mean : 2098.297
##  ARIMA(5,0,1) with non-zero mean : 2103.874
##  ARIMA(4,0,0) with zero mean     : 2089.178
##  ARIMA(3,0,0) with zero mean     : 2161.152
##  ARIMA(5,0,0) with zero mean     : 2093.927
##  ARIMA(4,0,1) with zero mean     : 2094.259
##  ARIMA(3,0,1) with zero mean     : 2102.707
##  ARIMA(5,0,1) with zero mean     : 2099.486
```

```
##
##  Best model: ARIMA(4,0,0) with zero mean
```

```
bic_model
```

```
## Series: train_data
## ARIMA(4,0,0) with zero mean
##
## Coefficients:
##           ar1      ar2      ar3      ar4
##       -0.0231  -0.4336  -0.4910  -0.4355
## s.e.   0.0466   0.0390   0.0388   0.0467
##
## sigma^2 estimated as 14.03:  log likelihood=-1029.76
## AIC=2069.53   AICc=2069.69   BIC=2089.18
```

For BIC, Best model is ARMA(4,0).

Choosing BEST Model

We will check MSE for both of these models. For whichever model the MSE is lowest, we will choose that.

```
pred_aic<- forecast(aic_model, h = 20)
aic_forecast<- pred_aic$mean
mse_aic = sum((test_data - aic_forecast)^2)/length(test_data)
print(paste("MSE for AIC Model = ", mse_aic))
```

```
## [1] "MSE for AIC Model =  37.4904858003665"
```

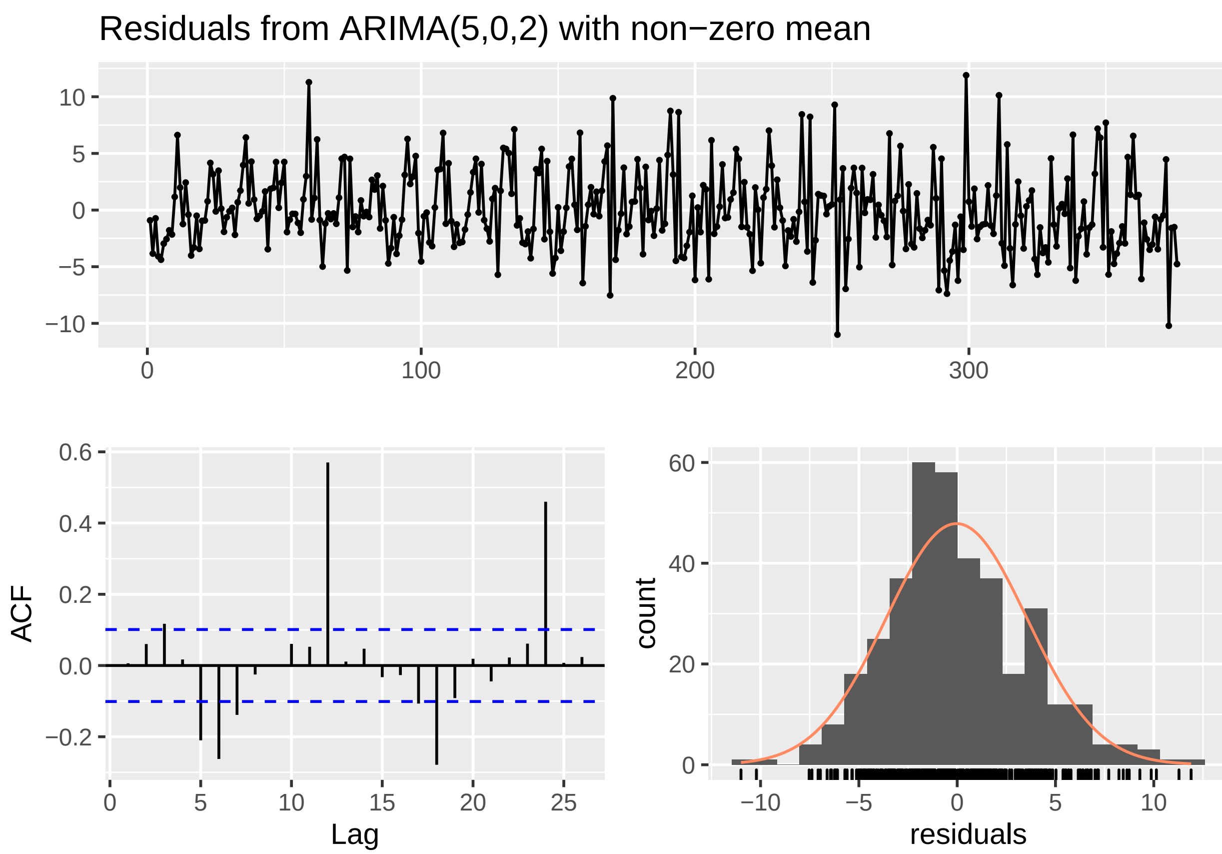
```
pred_bic <- forecast(bic_model, h = 20)
bic_forecast<- pred_bic$mean
mse_bic = sum((test_data - bic_forecast)^2)/length(test_data)
print(paste("MSE for BIC Model = ", mse_bic))
```

```
## [1] "MSE for BIC Model =  38.2007743700181"
```

As we can see, MSE for AIC model is lowest. So we will choose that. Therefore ARMA(5,2) is the best model and we will forecast with respect to this model.

Residuals

```
checkresiduals(aic_model)
```

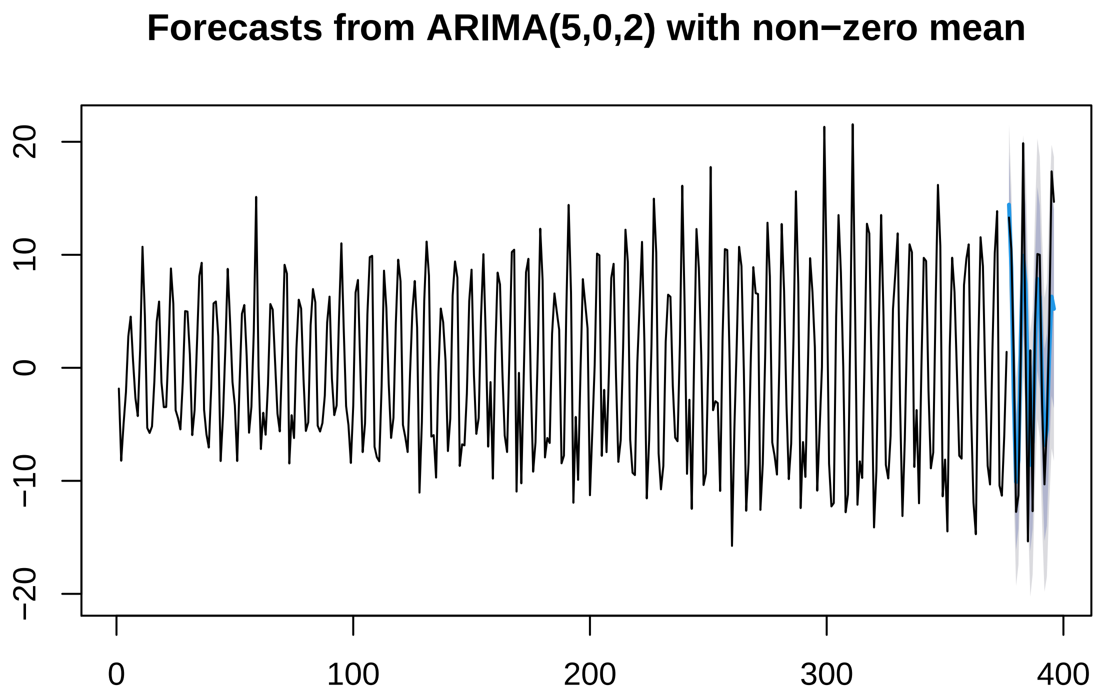



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,2) with non-zero mean
## Q* = 60.265, df = 3, p-value = 5.159e-13
##
## Model df: 8.   Total lags used: 11
```

We can see that residual follows almost a normal distribution which is a good sign for this model.

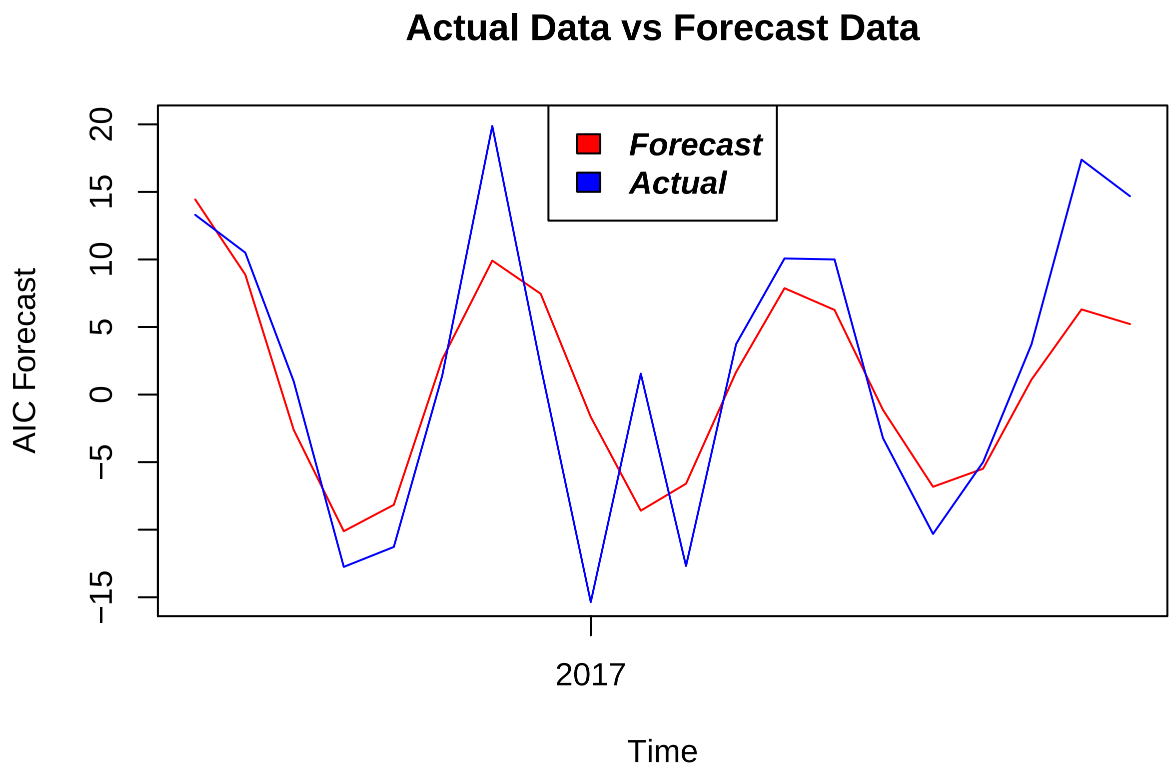
Forecasting

```
plot(forecast(aic_model,h=20),type='l')
lines(c((train_size+1):n),test_data, col = 'black')
```




```r
plot(Time[(train_size  +1): n],aic_forecast,type='l',col = 'red',
     ylim = c(-15,20),xlab = 'Time',ylab = 'AIC Forecast',
     main = 'Actual Data vs Forecast Data')

lines(Time[(train_size+1):n],test_data, col = 'blue')
legend('top', legend=c("Forecast", "Actual"),fill=c("red", "blue"), text.font=4)
```



Inverse Transformation

```
y_hat = rep(0, (length(test_data) + 1))
y_hat[1] = data[(length(train_data))]
for (i in 2:(length(test_data) + 1)) {
  y_hat[i] = aic_forecast[i - 1] +  y_hat[i - 1]
}

df_compare = data.frame(cbind(y_hat[2:(length(test_data) + 1)], data[c((train_size+2) : length(Time))])
## Took (train_size + 2), since in the main data we have one extra row, first one.
colnames(df_compare) = c("Predicted", "Actual")
df_compare
```

```
##     Predicted   Actual
## 1  103.76434 104.0375
## 2  112.64052 114.5397
## 3  110.02710 115.5159
## 4   99.91560 102.7637
## 5   91.75646  91.4867
## 6   94.36664  92.8900
## 7  104.28110 112.7694
## 8  111.73824 114.8505
## 9  110.07556  99.4901
## 10 101.48482 101.0396
## 11  94.88823  88.3530
```

```
## 12  96.55488  92.0805
## 13 104.42672 102.1532
## 14 110.69114 112.1538
## 15 109.55664 108.9312
## 16 102.73284  98.6154
## 17  97.24516  93.6137
## 18  98.35416  97.3359
## 19 104.65782 114.7212
## 20 109.87463 129.4048
```

Now let's plot it.

```r
plot(Time[(train_size+2):length(Time)],df_compare$Predicted,type='l',col = 'red', ylim = c(80,130),
     ylab = 'Eletric Production',xlab ='Time',
     main = 'Actual Data vs Forecast Data for Main Data')

lines(Time[(train_size+2):length(Time)],df_compare$Actual, col = 'blue')
legend('bottomright', legend=c("Forecast", "Actual"),fill = c("red", "blue"), text.font=4)
```