Time Series Analysis On Global Temperature Anomaly

Group Number: 58

First Name	Last Name	Online Students? (Y or N)	Monday or Tuesday	Shared with ITMD 525? (Y or N)
Roshni	Jariwala	N	Tuesday	N
Manthan	Kapadia	N	Tuesday	N
Arpitha	Jayarama	N	Tuesday	N

Table of Contents

1. Introduction and Motivations

Climate change for many countries in the world is one of the biggest environmental threats to food production, water availability, forest biodiversity and livelihoods. Although any individual extreme climate event cannot be attributed unequivocally to climate change, the probability of high temperature events will increase if there is an underlying trend of rising mean temperature. The number of climate-related disasters has increased significantly since 1970's, hence performing time series analysis of weather data can be a very valuable tool to investigate its variability pattern and to predict short- and long-term changes in the time series.

The aim of this project is to perform time series analysis on the global temperature evolution. Time series analysis and forecasting has become a major tool in numerous applications in meteorology and other environmental areas to understand phenomena, like rainfall, humidity, temperature, draught etc. Time series is one of the interesting topics in data analytics, we are motivated towards exploring this method of analysis where we are also able to perform future predictions. The approach is to work on the data based on land-ocean and land-air data which involves analyzing both the areas and coming up with the best model which can predict overall temperature.

2. Data Description

Our data set is regarding the temperature anomaly that is recorded monthly over a span of 137 years i.e. 1880 to 2017. We collected the data set from National Centers for Environmental Information, it contains about 1646 observations including features like temperature anomaly, total error variance, high and low frequency error variance, bias error variance and diagnostic variable.

Where:
Temperature anomaly: It is the deviation of the temperature from the reference value or long term average. positive anomaly means that the temperature is higher/warmer than the reference value and negative anomaly means the temperature is lesser/cooler than the reference value.

Total error variance: It is the error generated while recording temperature. This is nothing but mean squared error.

3. Research Problems and Solutions

We chose the subject as global temperature changes as it is one of the major concern in the world right now, there have been scenario's where people were not prepared for this abrupt change in the climatic conditions hence we think it is important to understand the behavior in the temperature changes and come up with predictions which can help people be prepared for the worst.

Assume that today is to a great degree warm, at some area. At that point, there is an inclination

for tomorrow to be hotter than normal. So, on general basis, whatever happens at some time in near future is influenced by what is happening or occurring right now.

So, the problem lies in analyzing the minute trends in temperature change which contains lot of influential information which helps to draw significant results about predicting the temperature.

A time series is any series of measurements taken at regular time intervals. Time series is the best available statistical methodology which can help us to analyze temperature behavior. So, we are trying to closely look into anomalies in temperature change and come up with a trend which will help in prediction of the exact future climatic conditions or near to it.

4. Model Learning

4.1. Data Processing

Our Data set consists of 1646 observations, with the below mentioned features:
year
month
anomaly of temperature (k)
total error variance (K**2)
high-frequency error variance (k**2)
low-frequency error variance (k**2)
bias error variance (K**2)
diagnostic variable
In this set of variables, we are choosing Year, month and anomaly of temperature as the analysis variable, as we are mainly focusing on time series analysis on anomaly of temperature. The Data is taken from NOAA (National Center for Environmental Information the link below: https://www.ncdc.noaa.gov/data-access/marineocean-data/noaa-global-surface-temperature-noaaglobaltemp

Files in Dataset Folder:
projectdata_train.txt
This is the dataset on monthly basis which we split into training and testing. Training dataset contains data till the year 1990. The data after 1990 is in testing dataset.

testdata.txt
This is the dataset on monthly basis which we split into training and testing. The data from 1991 is in testing dataset which we will be using for evaluation purpose.

testingprojectdata.txt
This is the dataset on yearly basis which we split into training and testing. The data from 1988 is in testing dataset which we will be using for evaluation purpose.

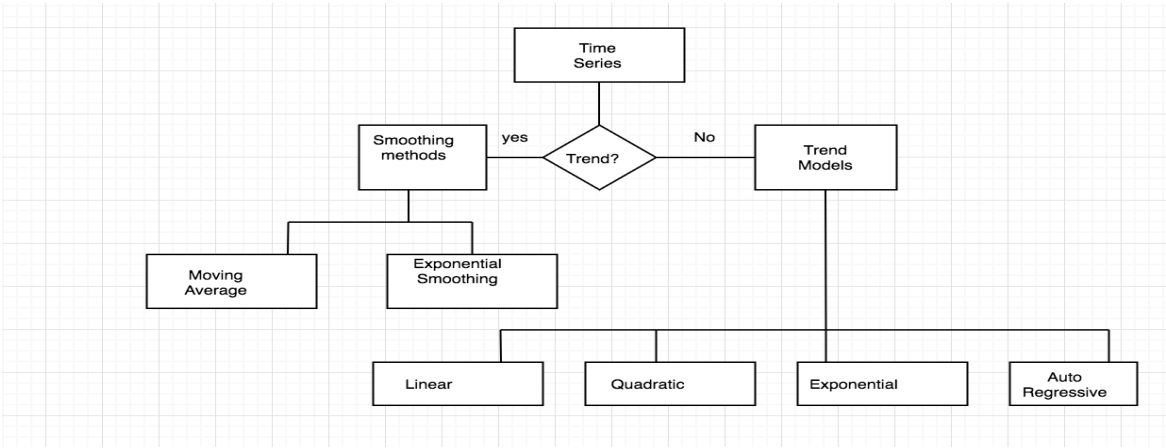
yearlyprojectdata.txt
This is the dataset on yearly basis which we split into training and testing. Training dataset contains data till the year 1987. The data after 1987 is in testing dataset.

Monthly Data Analysis

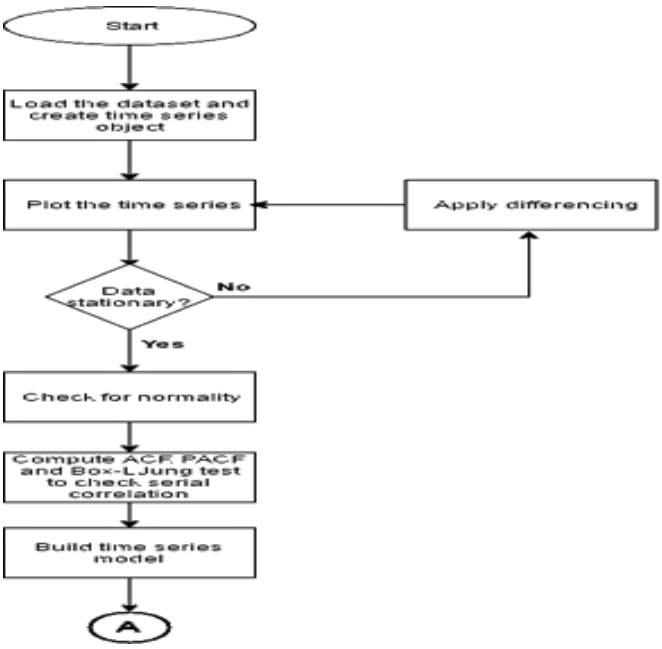
Normal time series analysis is performed on the data to understand the time series. During the initial findings, we applied differencing on the data to make the time series stationary and serially correlated.

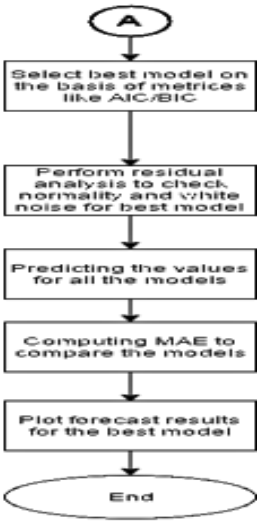
Initially we tried to build AR, MA, ARIMA and ARMA models but then none of the models produced effective results. Hence, we investigated more regarding the trends of the time series.

As we can see from the below plot there's almost no trend in the data. When there's no trend, we go for exponential smoothing techniques.



4.2. Data Analytics Tasks and Processes





ANALYSIS ON THE BASIS OF YEARLY DATA

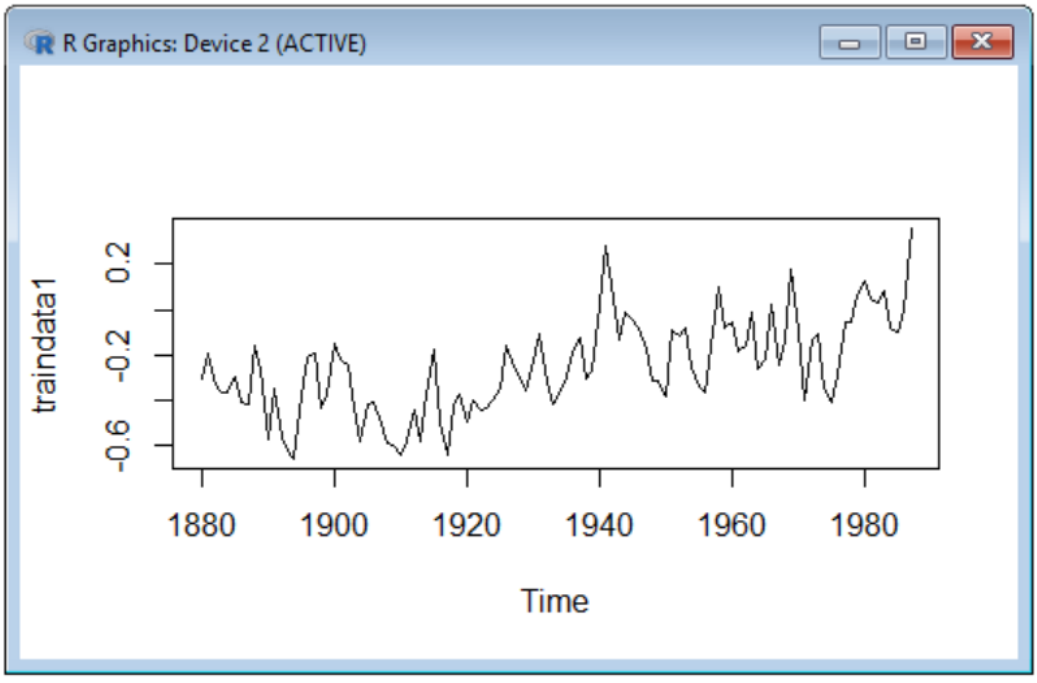
1. Load the data set and create the time series object:

```
> # Read the table
> tdata= read.table("yearlyprojectdata.txt",header=T)
> tdata
    year Anomalyintemp
1   1880  -0.304566333
2   1881  -0.189088500
3   1882  -0.332050417
4   1883  -0.361608000
5   1884  -0.366577333
6   1885  -0.294791167
7   1886  -0.409250500
8   1887  -0.416872167
9   1888  -0.155700917
```

```
>  atemp1=tdata$Anomalyintemp
> # creating time series object
> traindata1=ts(atemp1 , start=c(1880,1), end=c(1987,1))
> traindata1
Time Series:
Start = 1880
End = 1987
Frequency = 1
  [1] -0.304566333 -0.189088500 -0.332050417 -0.361608000 -0.366577333
  [6] -0.294791167 -0.409250500 -0.416872167 -0.155700917 -0.264441167
 [11] -0.569495167 -0.343048167 -0.560641167 -0.619818083 -0.655651167
 [16] -0.384822667 -0.205954583 -0.189602083 -0.431098167 -0.376680333
 [21] -0.146303750 -0.220319000 -0.246180250 -0.435840500 -0.576463000
 [26] -0.418637417 -0.410246250 -0.505824417 -0.589863500 -0.599565583
 [31] -0.636921667 -0.583191750 -0.441634000 -0.582710417 -0.369924583
 [36] -0.175123667 -0.496508167 -0.636753750 -0.422593500 -0.376348917
 [41] -0.495156833 -0.396822417 -0.440604917 -0.434557000 -0.397305917
 [46] -0.351406167 -0.159385417 -0.250795000 -0.295448833 -0.359087750
 [51] -0.230924250 -0.102405667 -0.313425250 -0.412616833 -0.371343750
 [56] -0.296801417 -0.191103083 -0.121604500 -0.308313167 -0.274056583
```

1. Plot the time series check for the stationarity of the data set.

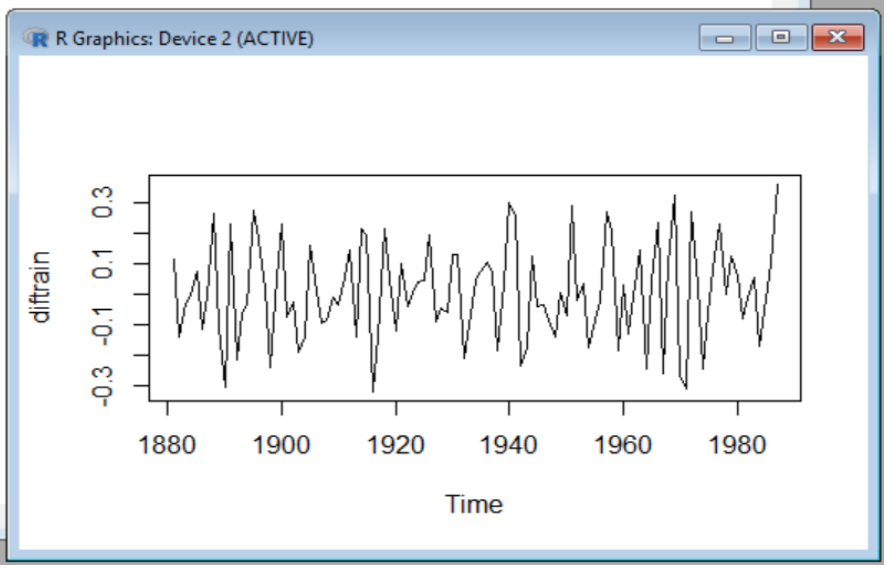
```
> # creating time plot
> plot(traindata1)
>
```



As we can observe from the graph that the time series is not stationary there is variation in mean and variance over the time.

3. Apply Differencing to make the data stationary plot the differenced time series object.

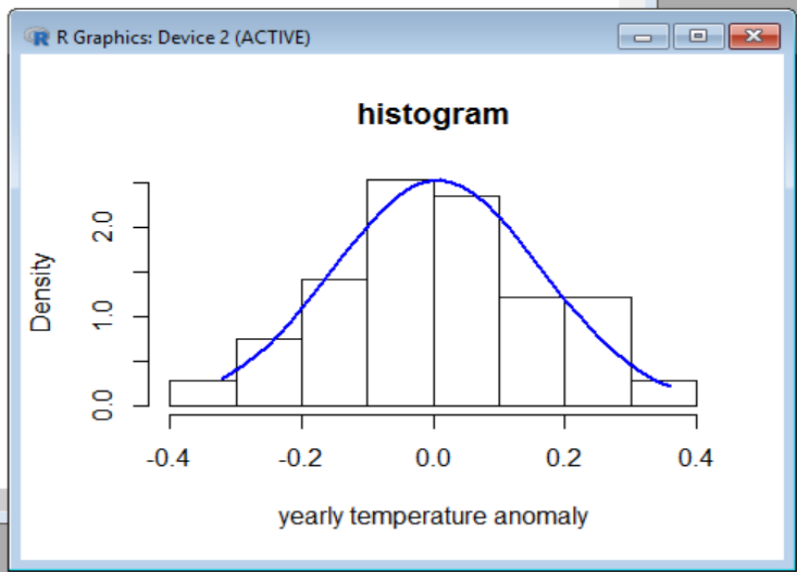
```
> #Applying differencing on the model
> diftrain=diff(traindata1)
> # plotting the differenced model
> plot(diftrain)
>
```



From the above plot, we can see that the data is stationary and therefore, we will not apply differencing again.

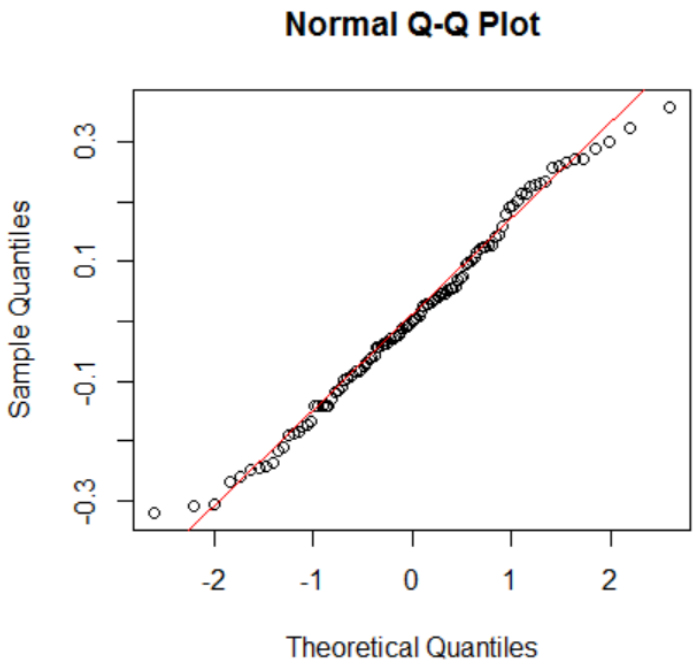
4. Normality check.

```
> # Creating the Histogram
> hist(diftrain, xlab="yearly temperature anomaly", prob=TRUE, main="histogram")
> xfit=seq(min(diftrain), max(diftrain), length=40)
> yfit=dnorm(xfit, mean=mean(diftrain), sd=sd(diftrain))
> lines(xfit, yfit, col="blue", lwd=2)
>
```



From the above histogram, we can see that the data has symmetric distribution.

```
> # Creating the QQ plot
> qqnorm(diftrain)
> qqline(coredata(diftrain), col=2)
>
```



From the above QQ plot, we can see that the data is normally distributed.

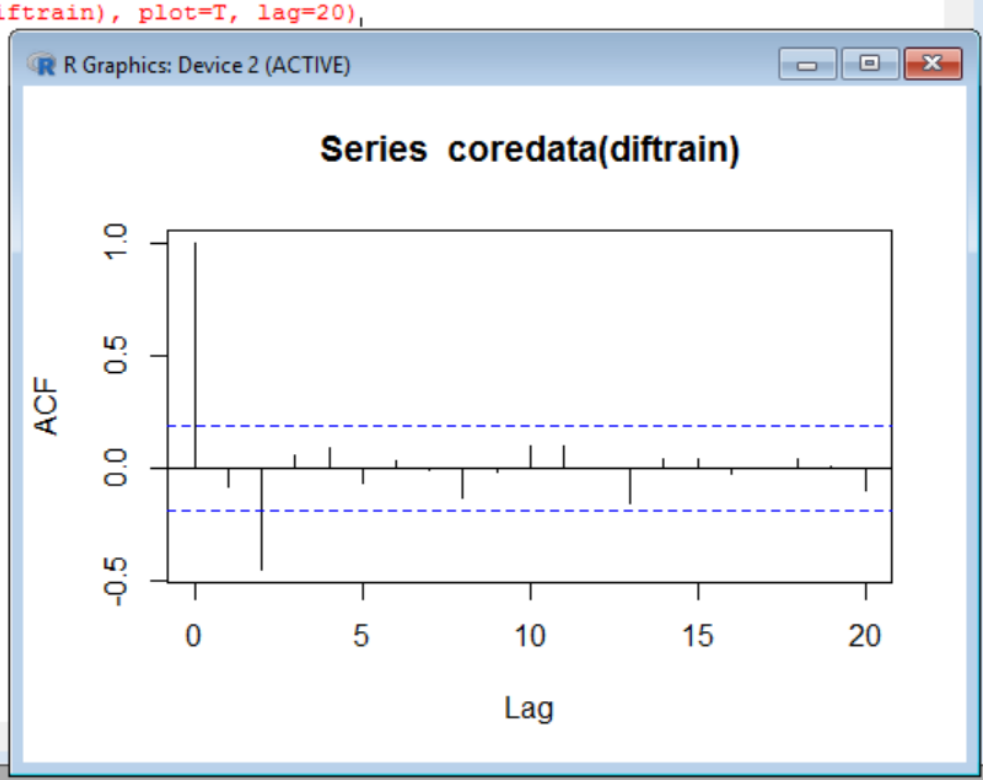
Performing Normality test.

```
> # Performing Normality Test
> normalTest(diftrain, method=c("jb"))

Title:
 Jarque - Bera Normalality Test

Test Results:
  STATISTIC:
    X-squared: 1.7555
  P VALUE:
    Asymptotic p Value: 0.4157

Description:
 Wed Apr 26 12:44:25 2017 by user: roshni
```

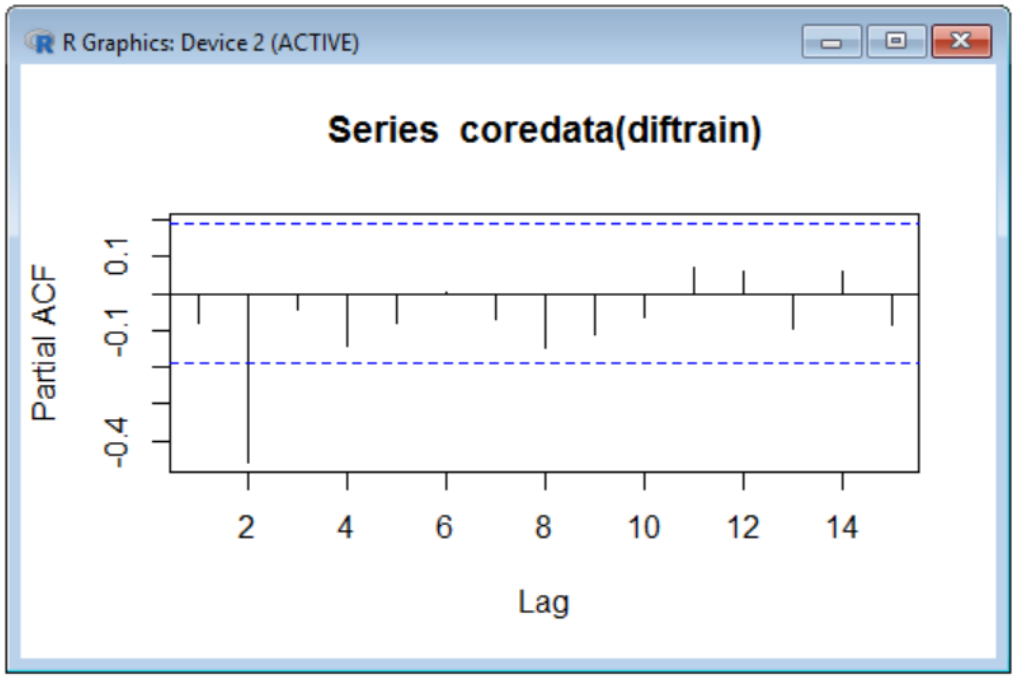
As we can observe that the p-value is greater than 0.05, we can say that the data follows normal distribution.

5. Plot ACF, PACF and Ljung-Box test to check serial correlation.

```
> # Computing and plotting the ACF
> acf(coredata(diftrain), plot=T, lag=20)
>
```




```
> # Computing and plotting the PACF
> pacf(coredata(diftrain), plot=T, lag=15)
>
```



As we can see that the data decays quickly, we can say that the data is serially correlated.
Computing Ljung-Box test.

```
> # Computing Ljung-Box test for lag 6 and 12
> Box.test(coredata(diftrain), lag=6, type='Ljung')

        Box-Ljung test

data:  coredata(diftrain)
X-squared = 24.832, df = 6, p-value = 0.0003669

> Box.test(coredata(diftrain), lag=12, type='Ljung')

        Box-Ljung test

data:  coredata(diftrain)
X-squared = 29.386, df = 12, p-value = 0.003451
```

As we can see that the p-value is less than 0.05 at 95% confidence interval. Therefore, we can say that the inflation rates are serially correlated.

6. Build Time series Models:

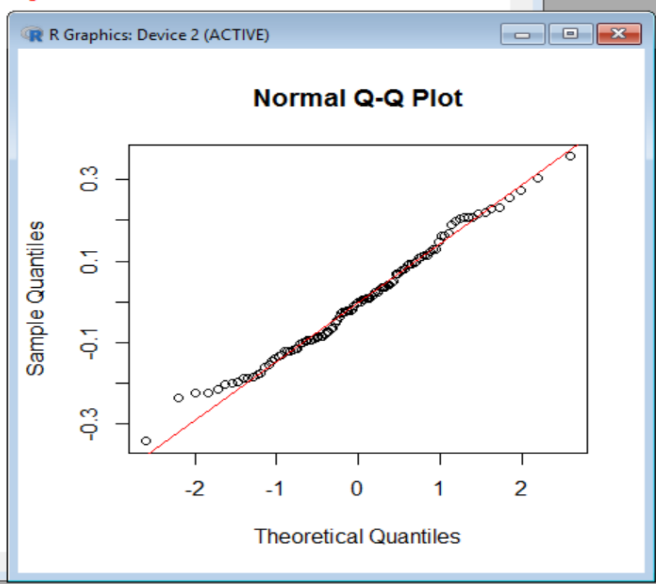
- **AR**

```
> #Computing the AR(2) model for yearly data
> yearlyAR=arima(diftrain, order=c(2,0,0))
> yearlyAR

Call:
arima(x = diftrain, order = c(2, 0, 0))

Coefficients:
          ar1      ar2  intercept
      -0.1284  -0.4803     0.0047
s.e.   0.0874   0.0868     0.0084

sigma^2 estimated as 0.01906:  log likelihood = 59.77,  aic = -111.53
```

Residual Analysis:

Computing QQplot for residual analysis.

```
> #computing QQ plot for residual analysis for AR model
> qqnorm(yearlyAR$residuals)
> qqline(yearlyAR$residuals, col=2)
>
```



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for AR model

```
> #computing Ljung Box test for residual analysis for AR model
> Box.test(yearlyAR$residuals, lag=6, type='Ljung')

        Box-Ljung test

data:  yearlyAR$residuals
X-squared = 3.0544, df = 6, p-value = 0.802

> Box.test(yearlyAR$residuals, lag=12, type='Ljung')

        Box-Ljung test

data:  yearlyAR$residuals
X-squared = 7.8261, df = 12, p-value = 0.7986
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

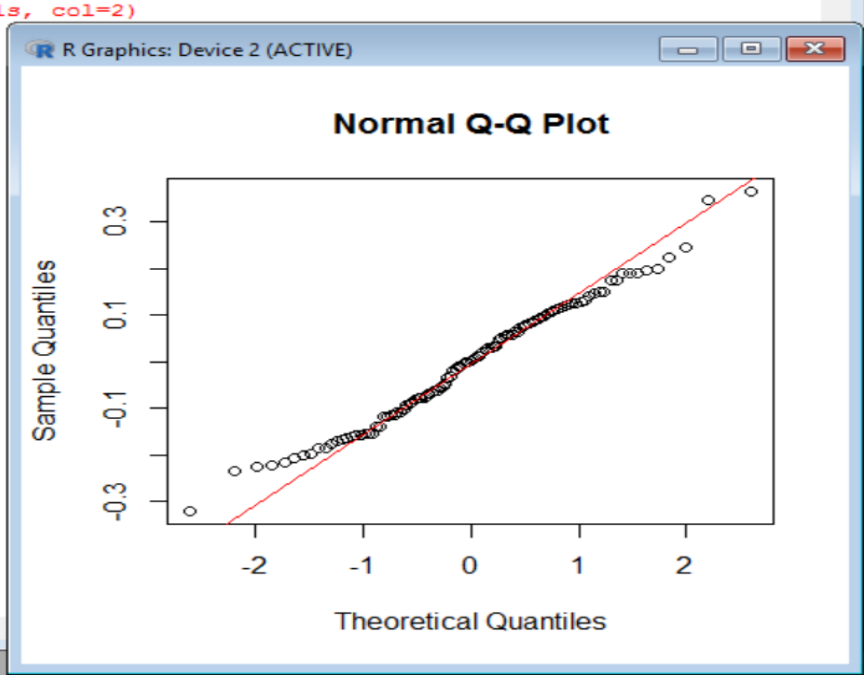
- **MA**

```
> #Computing the MA(2) model for yearly data
> yearlyMA=arima(diftrain, order=c(0,0,2))
> yearlyMA

Call:
arima(x = diftrain, order = c(0, 0, 2))

Coefficients:
          ma1      ma2  intercept
      -0.2362  -0.5726     0.0040
s.e.   0.0856   0.0884     0.0027

sigma^2 estimated as 0.01786:  log likelihood = 62.94,  aic = -117.87
```

Residual Analysis:

Computing QQplot for residual analysis.

```
> #computing QQ plot for residual analysis for MA model
> qqnorm(yearlyMA$residuals)
> qqline(yearlyMA$residuals, col=2)
>
```



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for MA model.

```
> #Computing Ljung Box test for residual analysis for MA model
> Box.test(yearlyMA$residuals, lag=6, type='Ljung')

        Box-Ljung test

data:  yearlyMA$residuals
X-squared = 1.582, df = 6, p-value = 0.9539

> Box.test(yearlyMA$residuals, lag=12, type='Ljung')

        Box-Ljung test

data:  yearlyMA$residuals
X-squared = 3.6979, df = 12, p-value = 0.9883
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

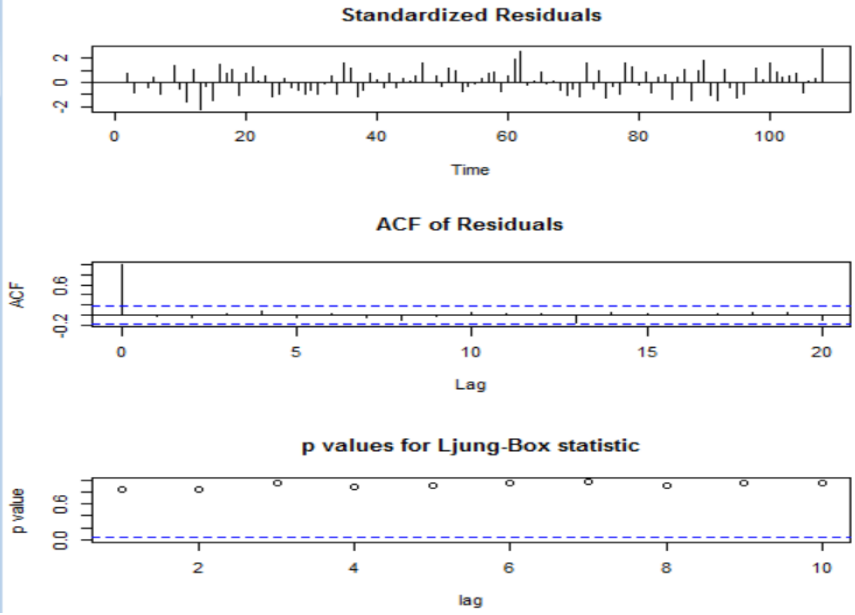
- **ARIMA**

```
> #Computing the ARIMA(p,d,q) model for yearly data using auto.arima function
> yearlyARIMA=auto.arima(coredata(traindata1), allowdrift=FALSE)
> yearlyARIMA
Series: coredata(traindata1)
ARIMA(0,1,2)

Coefficients:
          ma1      ma2
      -0.2071  -0.5414
s.e.   0.0836   0.0820

sigma^2 estimated as 0.01854:  log likelihood=62.08
AIC=-118.16   AICc=-117.93   BIC=-110.15
```

Residual Analysis:

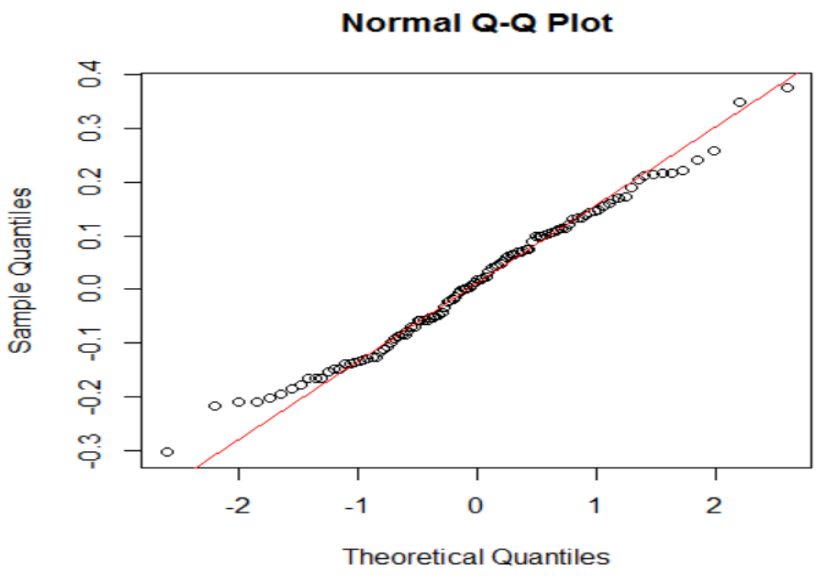
Computing residual analysis for the ARIMA(p,d,q) model

```
> tsdiag(yearlyARIMA)
>
```



Computing Q-Q plot for residual analysis for ARMA(0,1,2) model.

```
> #Plotting the QQ plot
> qqnorm(res)
> qqline(res, col=2)
>
```



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for ARMA(2,0,2) model.

```
> #Performing Ljung Box test for residuals
> Box.test(yearlyARIMA$residuals, lag=6, type='Ljung')

        Box-Ljung test

data:  yearlyARIMA$residuals
X-squared = 1.5929, df = 6, p-value = 0.9531

> Box.test(yearlyARIMA$residuals, lag=12, type='Ljung')

        Box-Ljung test

data:  yearlyARIMA$residuals
X-squared = 4.0482, df = 12, p-value = 0.9826
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

- **ARMA**

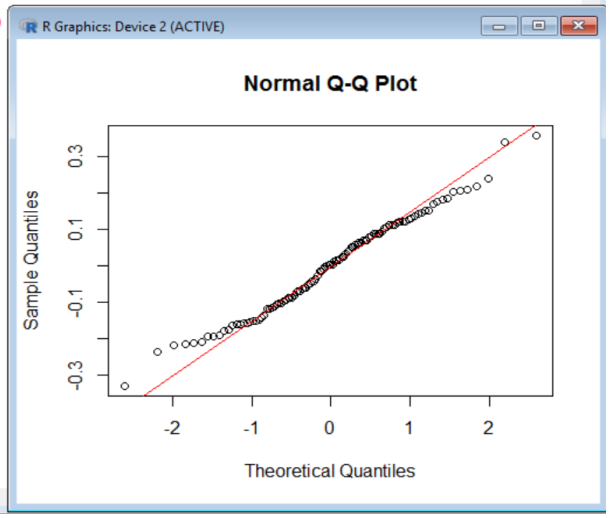
a) Building the ARMA model using arima function.

```
> #Computing ARMA(2,2) model for yearly data
> yearlyARMA=arima(diftrain, order=c(2,0,2), method='ML',include.mean=T)
> yearlyARMA

Call:
arima(x = diftrain, order = c(2, 0, 2), include.mean = T, method = "ML")

Coefficients:
         ar1      ar2      ma1      ma2  intercept
      0.0447  -0.0808  -0.2729  -0.5035      0.004
s.e.  0.1868   0.1777   0.1684   0.1772      0.003

sigma^2 estimated as 0.01783:  log likelihood = 63.04,  aic = -114.09
```

Residual Analysis:

Computing Q-Q plot for residual analysis for ARMA (2,0,2) model.

```
> #Computing QQ plot for residual analysis for ARMA(2,0,2) model built using arima function
> qqnorm(yearlyARMA$residuals)
> qqline(yearlyARMA$residuals, col=2)
>
```



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for ARMA (2,0,2) model.

```
> #Computing Ljung Box test for residual analysis for ARMA(2,0,2) model
> Box.test(yearlyARMA$residuals, lag=6, type='Ljung')

        Box-Ljung test

data:  yearlyARMA$residuals
X-squared = 1.0885, df = 6, p-value = 0.982

> Box.test(yearlyARMA$residuals, lag=12, type='Ljung')

        Box-Ljung test

data:  yearlyARMA$residuals
X-squared = 3.4344, df = 12, p-value = 0.9916
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

b) Computing ARMA model using EACF function.

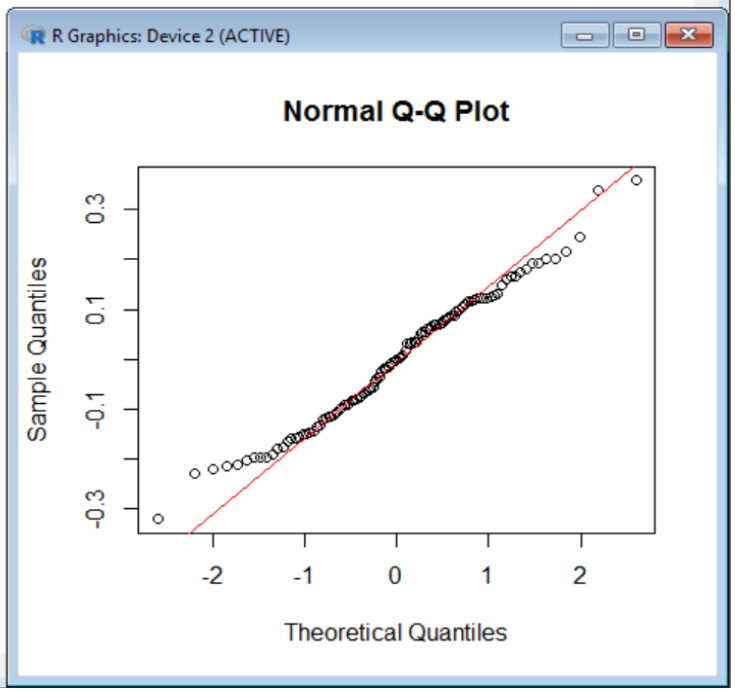
```
> # computing ARMA(p,q) model using EACF function
> EACF(diftrain)
[1] "EACF table"
      [,1]  [,2]   [,3]   [,4]    [,5]    [,6]
[1,] -0.08 -0.45  0.055  0.090 -0.0650  0.0356
[2,] -0.18 -0.46  0.211  0.101 -0.0111  0.0284
[3,] -0.12 -0.29 -0.162  0.107 -0.0396  0.0097
[4,] -0.32 -0.24 -0.160  0.154  0.0034  0.0106
[5,] -0.38  0.14 -0.227 -0.061 -0.0777 -0.1449
[6,] -0.13 -0.16 -0.139 -0.020 -0.0047 -0.1854
[1] " "
[1] "Simplified EACF: 2 denotes significance"
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    2    0    0    0    0
[2,]    0    2    2    0    0    0
[3,]    0    2    0    0    0    0
[4,]    2    2    0    0    0    0
[5,]    2    0    2    0    0    0
[6,]    0    0    0    0    0    0
```

```
> #computing ARMA(1,3) model for yearly data
> yearlyARMA_eacf=arima(diftrain, order=c(1,0,3), method='ML', include.mean=T)
> yearlyARMA_eacf

Call:
arima(x = diftrain, order = c(1, 0, 3), include.mean = T, method = "ML")

Coefficients:
          ar1     ma1      ma2      ma3  intercept
      -0.8288  0.6148  -0.7637  -0.5184     0.0040
s.e.   0.4014  0.3952   0.1347   0.2211     0.0025

sigma^2 estimated as 0.01778:  log likelihood = 63.15,  aic = -114.31
```

Residual Analysis:

Computing QQplot for residual analysis for ARMA(2,0,2) model.

```
> #Computing QQ plot for residual analysis for ARMA(1,0,3) model built using eacf function
> qqnorm(yearlyARMA_eacf$residuals)
> qqline(yearlyARMA_eacf$residuals, col=2)
>
```



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for ARMA(1,0,2) model.

```
> #Computing Ljung Box test for residual analysis for ARMA(1,0,3) model
> Box.test(yearlyARMA_eacf$residuals, lag=6, type='Ljung')

        Box-Ljung test

data:  yearlyARMA_eacf$residuals
X-squared = 1.1391, df = 6, p-value = 0.9798

> Box.test(yearlyARMA_eacf$residuals, lag=12, type='Ljung')

        Box-Ljung test

data:  yearlyARMA_eacf$residuals
X-squared = 3.3342, df = 12, p-value = 0.9927
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

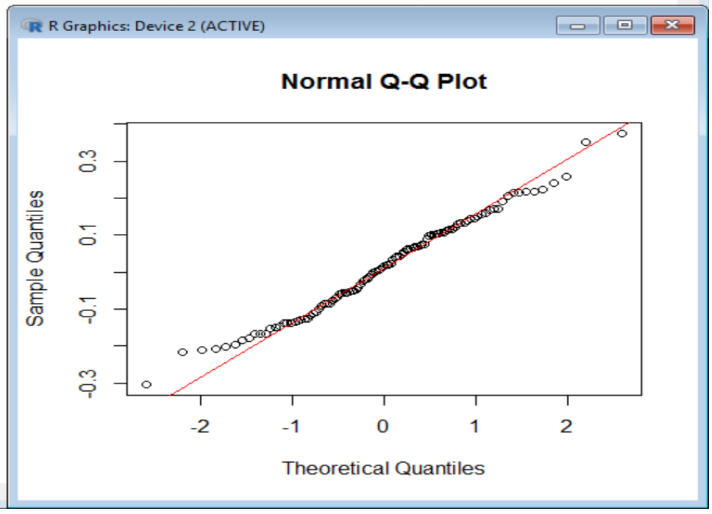
c) Building the ARMA model using auto arima function.

```
> #Computing ARMA(p,q) model for yearly data using auto.arima function
> yearlyARMA1=auto.arima(diftrain, max.P=8, max.Q=8, ic="aic")
> yearlyARMA1
Series: diftrain
ARIMA(0,0,2) with zero mean

Coefficients:
          ma1      ma2
      -0.2071  -0.5414
s.e.   0.0836   0.0820

sigma^2 estimated as 0.01854:  log likelihood=62.08
AIC=-118.16   AICc=-117.93   BIC=-110.15
```

Residual Analysis:

Computing QQplot for residual analysis for ARMA(0,0,2) model.

```
> #Computing QQ plot for residual analysis for ARMA(0,0,2) model built using auto.arima function
> qqnorm(yearlyARMA1$residuals)
> qqline(yearlyARMA1$residuals, col=2)
>
```



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for ARMA(0,0,2) model.

```
> #Computing Ljung Box test for residual analysis for ARMA(0,0,2) model.
> Box.test(yearlyARMA1$residuals, lag=6, type='Ljung')

        Box-Ljung test

data:  yearlyARMA1$residuals
X-squared = 1.5696, df = 6, p-value = 0.9547

> Box.test(yearlyARMA1$residuals, lag=12, type='Ljung')

        Box-Ljung test

data:  yearlyARMA1$residuals
X-squared = 3.9513, df = 12, p-value = 0.9843
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

7. Comparing the best model on the basis of AIC value.

Using the AIC value as the criteria , we can observe that the best model is ARIMA(p,d,q) model that is got by arima function.

8. Loading the testing data to calculate the MAE values.

```
> #Loading the testing data
> testdata=read.table("testingprojectdata.txt", header=T)
> observedatemp=testdata$Anomalyintemp
> observedatemp
 [1]  0.179757500 -0.034993667  0.168784500  0.153242417  0.005804417
 [6]  0.124980500  0.106543917  0.225135333  0.158133417  0.324352500
[11]  0.475151917  0.137129583  0.134408333  0.316912500  0.389063833
[16]  0.440264333  0.416532750  0.446083750  0.405578917  0.337626917
[21]  0.250932500  0.508649083  0.593459583  0.283305000  0.353285667
[26]  0.435618417  0.573686750  0.778798333  0.782137250  0.669092500
```

9. Predicting the values for the models.

- Predicting the value for the AR (2) model

```
> #predicting the value for AR model
> ar_predict=predict(yearlyAR, n.ahead=30, se.fit=T)
> ar_predict
$pred
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] -0.086601232 -0.153997782  0.068926398  0.072666404 -0.034878155
 [6] -0.022863141  0.027244497  0.015039030 -0.007458802  0.001292390
[11]  0.010973613  0.005527365  0.001577170  0.004700148  0.006196253
[16]  0.004504239  0.004002995  0.004879994  0.005008100  0.004570450
[21]  0.004565129  0.004776003  0.004751478  0.004653350  0.004677731
[26]  0.004721728  0.004704368  0.004685467  0.004696232  0.004703927

$se
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] 0.1380699 0.1392038 0.1532255 0.1541371 0.1567688 0.1572057 0.1576810
 [8] 0.1578451 0.1579228 0.1579758 0.1579866 0.1580020 0.1580031 0.1580072
[15] 0.1580073 0.1580083 0.1580083 0.1580085 0.1580085 0.1580086 0.1580086
[22] 0.1580086 0.1580086 0.1580086 0.1580086 0.1580086 0.1580086 0.1580086
[29] 0.1580086 0.1580086
```

- Predicting the value for the MA (2) model

```
> #Predicting the value for MA(2) model
> ma_predict=predict(yearlyMA, n.ahead=30, se.fit=T)
> ma_predict
$pred
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] -0.098382372 -0.204508068  0.003996876  0.003996876  0.003996876
 [6]  0.003996876  0.003996876  0.003996876  0.003996876  0.003996876
[11]  0.003996876  0.003996876  0.003996876  0.003996876  0.003996876
[16]  0.003996876  0.003996876  0.003996876  0.003996876  0.003996876
[21]  0.003996876  0.003996876  0.003996876  0.003996876  0.003996876
[26]  0.003996876  0.003996876  0.003996876  0.003996876  0.003996876

$se
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] 0.1336477 0.1373242 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079
 [8] 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079
[15] 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079
[22] 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079 0.1572079
[29] 0.1572079 0.1572079
```

- Predicting the value for the ARIMA (2,0,2) model

```
> #Predicting the value for ARIMA(p,d,q) model
> arima_predict=predict(yearlyARIMA, n.ahead=30, se.fit=T)
> arima_predict
$pred
Time Series:
Start = 109
End = 138
Frequency = 1
 [1] 0.25826094 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231
 [7] 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231
[13] 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231
[19] 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231
[25] 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231 0.05499231

$se
Time Series:
Start = 109
End = 138
Frequency = 1
 [1] 0.1361469 0.1737525 0.1770943 0.1803741 0.1835954 0.1867611 0.1898740
 [8] 0.1929367 0.1959516 0.1989208 0.2018463 0.2047300 0.2075736 0.2103788
[15] 0.2131471 0.2158799 0.2185785 0.2212442 0.2238782 0.2264815 0.2290553
[22] 0.2316004 0.2341179 0.2366086 0.2390734 0.2415130 0.2439282 0.2463197
[29] 0.2486883 0.2510344
```

- Predicting the ARMA (2,2) model

```
> #Predicting the value for ARMA(2,2) model
> arma_predict=predict(yearlyARMA, n.ahead=30, se.fit=T)
> arma_predict
$pred
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] -0.096802025 -0.208935922  0.002622423  0.021150267  0.004880423
 [6]  0.002655057  0.003870472  0.004104709  0.004016955  0.003994097
[11]  0.004000167  0.004002286  0.004001891  0.004001702  0.004001725
[16]  0.004001741  0.004001740  0.004001739  0.004001739  0.004001739
[21]  0.004001739  0.004001739  0.004001739  0.004001739  0.004001739
[26]  0.004001739  0.004001739  0.004001739  0.004001739  0.004001739

$se
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] 0.1335472 0.1369786 0.1583245 0.1583282 0.1584563 0.1584567 0.1584575
 [8] 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575
[15] 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575
[22] 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575 0.1584575
[29] 0.1584575 0.1584575
```

- Predicting the ARMA model built through EACF function.

```
> #predicting the value of ARMA(p,q) model built through EACF function
> eacfarma_predict=predict(yearlyARMA_eacf, n.ahead=30, se.fit=T)
> eacfarma_predict
$pred
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] -0.0985506197 -0.2018433927 -0.0109548877  0.0163664755 -0.0062782007
 [6]  0.0124903123 -0.0030655330  0.0098275674 -0.0008585789  0.0079983848
[11]  0.0006574965  0.0067418210  0.0016989702  0.0058786198  0.0024144145
[16]  0.0052856404  0.0029058916  0.0048782910  0.0032435137  0.0045984609
[21]  0.0034754444  0.0044062305  0.0036347701  0.0042741771  0.0037442195
[26]  0.0041834625  0.0038194062  0.0041211458  0.0038710559  0.0040783371

$se
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] 0.1333395 0.1363606 0.1571770 0.1572367 0.1572777 0.1573059 0.1573252
 [8] 0.1573385 0.1573476 0.1573539 0.1573582 0.1573612 0.1573632 0.1573646
[15] 0.1573656 0.1573662 0.1573667 0.1573670 0.1573672 0.1573673 0.1573674
[22] 0.1573675 0.1573676 0.1573676 0.1573676 0.1573676 0.1573676 0.1573676
[29] 0.1573676 0.1573676
```

- Predicting the ARMA model built using auto.arima function.

```
> #Predicting the value for ARMA(p,q) model using auto.arima function
> arma_predict1=predict(yearlyARMA1, n.ahead=30, se.fit=T)
> arma_predict1
$pred
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] -0.1003626 -0.2032686  0.0000000  0.0000000  0.0000000  0.0000000
 [7]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
[13]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
[19]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
[25]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000

$se
Time Series:
Start = 1988
End = 2017
Frequency = 1
 [1] 0.1361469 0.1390354 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671
 [8] 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671
[15] 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671
[22] 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671 0.1573671
[29] 0.1573671 0.1573671
```

Loading the data with the observed data and the removed prediction value along with the prediction value after removing the difference.

```
> #Loading the new data with predicted values in it
> predictionvalue=read.table("predictiondata.csv", header=T, sep=',')
> observedatemp=predictionvalue$Anomalyintemp
> observedatemp
```

10. Compute MAE value for the models.

```
> #Computing the MAE value for AR(2) model
> predictedar=predictionvalue[,4]
> mae_ar=sum(abs(observedatemp-predictedar))/nrow(testdata)
> mae_ar
[1] 0.111604
> #Computing the MAE value for MA(2) model
> predictedma=predictionvalue[,6]
> mae_ma=sum(abs(observedatemp-predictedma))/nrow(testdata)
> mae_ma
[1] 0.1080877
> #Computing the MAE value for ARIMA(p,d,q) model
> predictedarima=predictionvalue[,8]
> mae_arima=sum(abs(observedatemp-predictedarima))/nrow(testdata)
> mae_arima
[1] 0.127175
> #Computing the MAE value for ARMA(2,0,2) model
> predictedarma=predictionvalue[,10]
> mae_arma=sum(abs(observedatemp-predictedarma))/nrow(testdata)
> mae_arma
[1] 0.1086772
> #Computing the MAE value for ARMA(1,0,3) model built using EACF
> predictedeacfarma=predictionvalue[,12]
> mae_eacfarma=sum(abs(observedatemp-predictedeacfarma))/nrow(testdata)
> mae_eacfarma
[1] 0.1077262
> #Computing the MAE value for ARMA(0,0,2) model built using auto.arima
> predictedarma1=predictionvalue[,14]
> mae_autoarma=sum(abs(observedatemp-predictedarma1))/nrow(testdata)
> mae_autoarma
[1] 0.108552
```

11. Select the best model based on the MAE value

From the above computation, we can see that the model ARMA (1,3) has the less MAE value than any other models. Therefore, we can say that the ARMA (1,3) model is the best model.

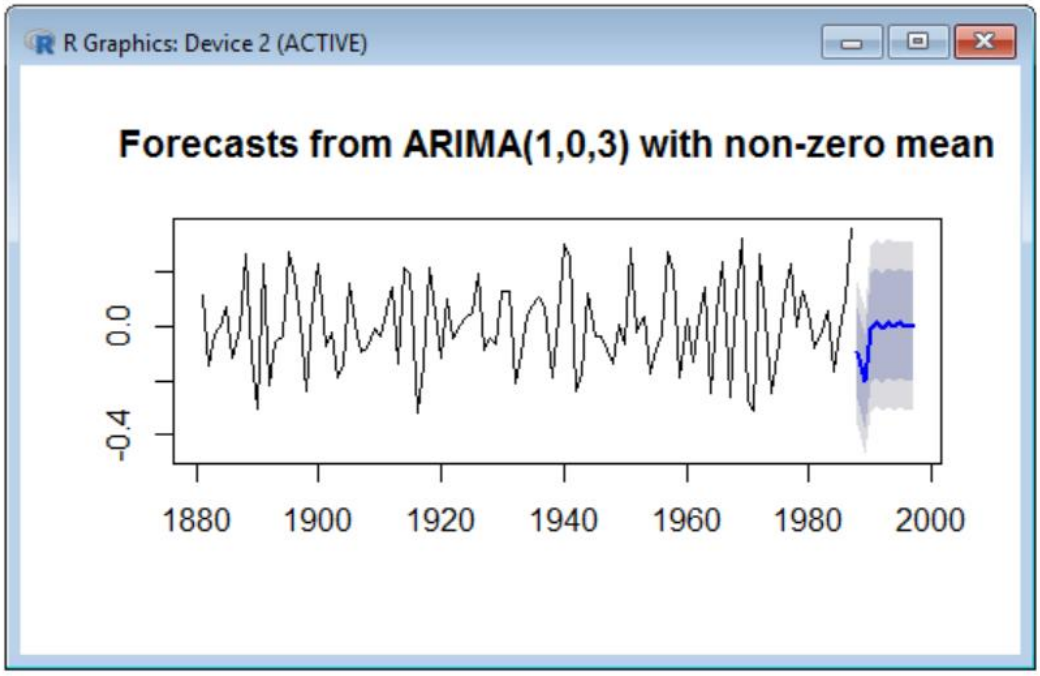
The equation for the best model is,

$r_t - \varphi_1 r_{t-1} = \varphi_0 + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \theta_3 at\text{-}3$

$r_t + 0.8288 r_{t-1} = \varphi_0 + a_t - 0.6148 a_{t-1} + 0.7637 a_{t-2} + 0.5184 a_{t-3}$

12. Plot Forecast Result for the best model.

```
> #Plotting the forecast to predict the future
```



ANALYSIS OF THE MONTHLY DATA

1. Remove the testing data set from the data.

2. Read the data after removing the values.

Read the table

> tdata= read.table("/Users/arpithajayarama/Desktop/projectdata.txt",header=T)

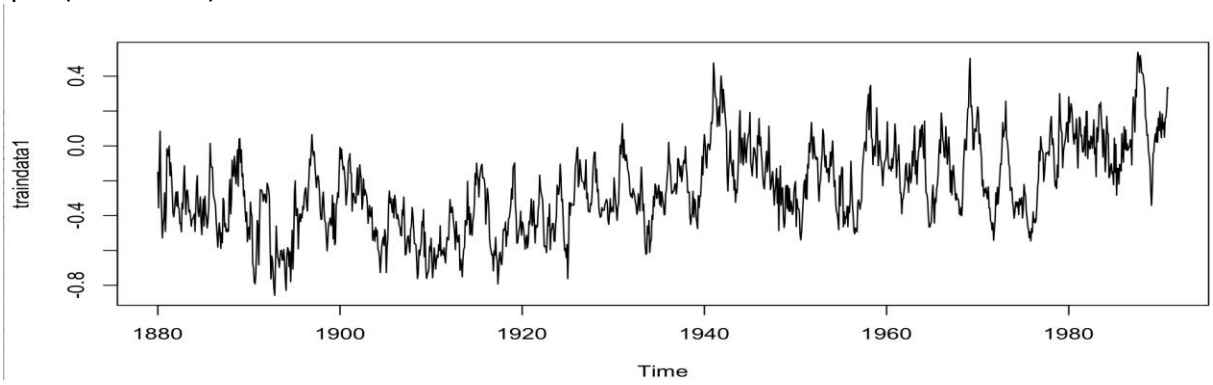
```
> tdata= read.table("/Users/arpithajayarama/Desktop/projectdata.txt",header=T)
> tdata
   Year Month Anamolyintemp TotalErrvar
1  1880     1      -0.150216    0.015989
2  1880     2      -0.355554    0.017320
3  1880     3      -0.099065    0.014291
4  1880     4       0.084517    0.015672
5  1880     5      -0.242055    0.014092
6  1880     6      -0.379139    0.014653
7  1880     7      -0.529209    0.014813
8  1880     8      -0.450185    0.014504
9  1880     9      -0.411194    0.014287
10 1880    10      -0.268624    0.015665
11 1880    11      -0.491696    0.014596
12 1880    12      -0.362376    0.014743
13 1881     1      -0.014747    0.016448
14 1881     2      -0.030257    0.017820
15 1881     3      -0.059735    0.014781
16 1881     4      -0.000145    0.016186
17 1881     5      -0.171720    0.014604
18 1881     6      -0.096571    0.015172
19 1881     7      -0.164791    0.015318
20 1881     8      -0.268421    0.014984
21 1881     9      -0.385346    0.014723
22 1881    10      -0.407760    0.016151
23 1881    11      -0.368138    0.015021
24 1881    12      -0.301431    0.015181
25 1882     1      -0.263983    0.016799
26 1882     2      -0.346597    0.018163
```

3. Create the time series object using ts function.

atemp1=tdata$Anamolyintemp
# creating time series object
> traindata1=ts(atemp1 , start=c(1880,1), end=c(1990,12),freq=12)

```
>
> atemp1=tdata$Anamolyintemp
> traindata1=ts(atemp1 , start=c(1880,1), end=c(1990,12),freq=12)
> traindata1
           Jan       Feb       Mar       Apr       May       Jun       Jul       Aug       Sep       Oct
1880 -0.150216 -0.355554 -0.099065  0.084517 -0.242055 -0.379139 -0.529209 -0.450185 -0.411194 -0.268624
1881 -0.014747 -0.030257 -0.059735 -0.000145 -0.171720 -0.096571 -0.164791 -0.268421 -0.385346 -0.407760
1882 -0.263983 -0.346597 -0.261965 -0.352866 -0.358269 -0.446409 -0.443634 -0.493581 -0.327213 -0.361921
1883 -0.267837 -0.394336 -0.262733 -0.251558 -0.302810 -0.381376 -0.366265 -0.398851 -0.461562 -0.425399
1884 -0.390346 -0.491358 -0.356029 -0.224149 -0.170127 -0.373882 -0.406454 -0.362280 -0.292036 -0.445124
1885 -0.305198 -0.460276 -0.287933 -0.384373 -0.407691 -0.471095 -0.434183 -0.360079 -0.211487  0.015706
1886 -0.122318 -0.269610 -0.302436 -0.313382 -0.350197 -0.439607 -0.505038 -0.581595 -0.503413 -0.441487
1887 -0.498543 -0.556702 -0.299459 -0.316439 -0.464846 -0.444460 -0.492323 -0.492967 -0.474490 -0.491347
1888 -0.391722 -0.306949 -0.080347 -0.165150 -0.175838 -0.059236 -0.136532 -0.218745 -0.119304 -0.020489
1889  0.042494 -0.095186 -0.021125 -0.177259 -0.159963 -0.288837 -0.256863 -0.367033 -0.435563 -0.380047
```

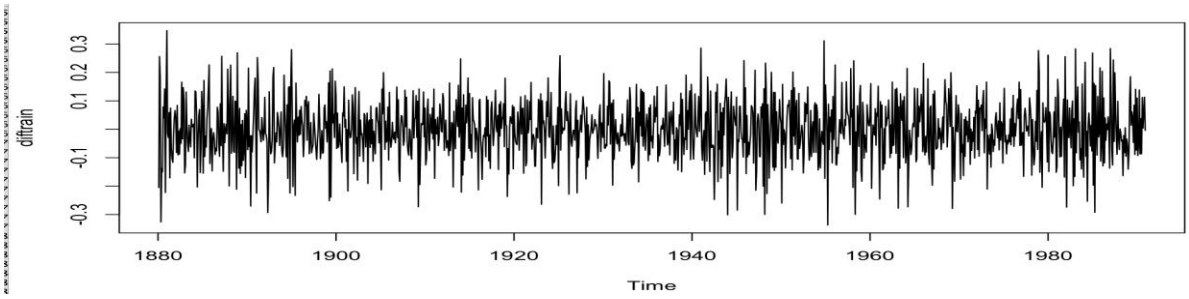
4. Plot the time series object for the mode.
plot(traindata1)



As we can observe from the graph that the time series is not stationary there is variation in mean and variance over the time.

5. Applying Differencing on the model to make it stationary.

> # Applying Dfferencing on the modle
> diftrain = diff(traindata1)
>
> # Plot the differenced model
> plot(diftrain)

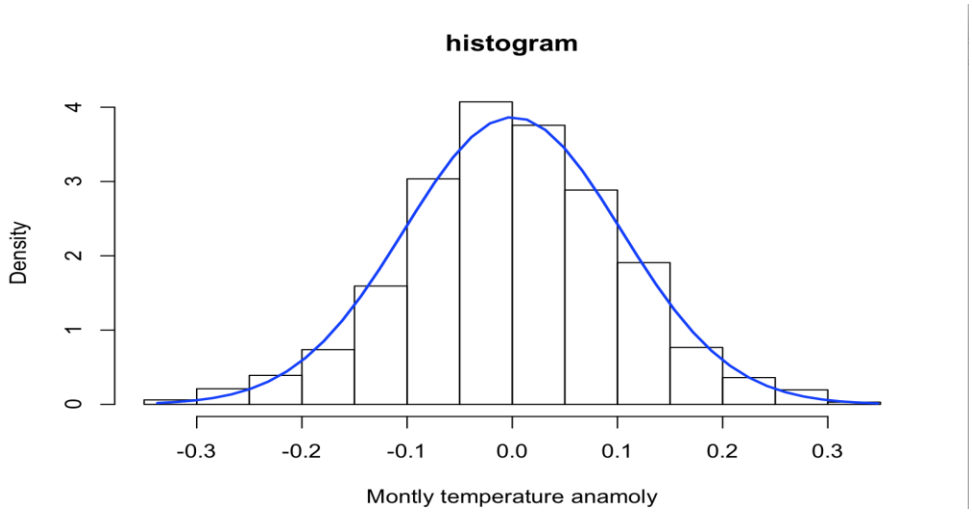


6. Get the Basic statistics for the differenced data.

```
> basicStats(diftrain)
                diftrain
nobs        1331.000000
NAs            0.000000
Minimum       -0.337310
Maximum        0.347629
1. Quartile   -0.064396
3. Quartile    0.070096
Mean           0.000362
Median        -0.000780
Sum            0.481516
SE Mean        0.002828
LCL Mean      -0.005187
UCL Mean       0.005911
Variance       0.010648
Stdev          0.103191
Skewness      -0.053219
Kurtosis       0.279342
```

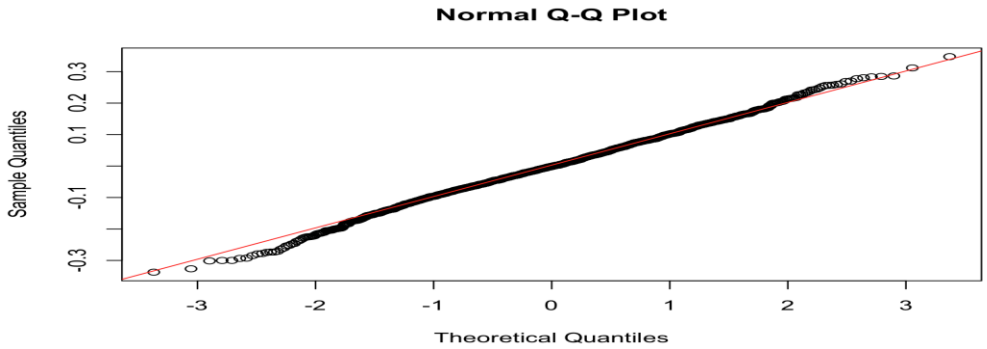
7. Plot the Histogram graph and check for the normality of the distribution.

> hist(diftrain, xlab="Montly temperature anamoly", prob=TRUE, main="histogram")
> xfit=seq(min(diftrain), max(diftrain), length=40)
> yfit=dnorm(xfit, mean=mean(diftrain), sd=sd(diftrain))
> lines(xfit, yfit, col="blue", lwd=2)



Form the plot we can observe that the data is normally distributed.

8. Plot the QQ plot to check the normality.



we can observe from the graph that most of the points lie in and around the normality line. hence we can conclude that the data is normally distributed.

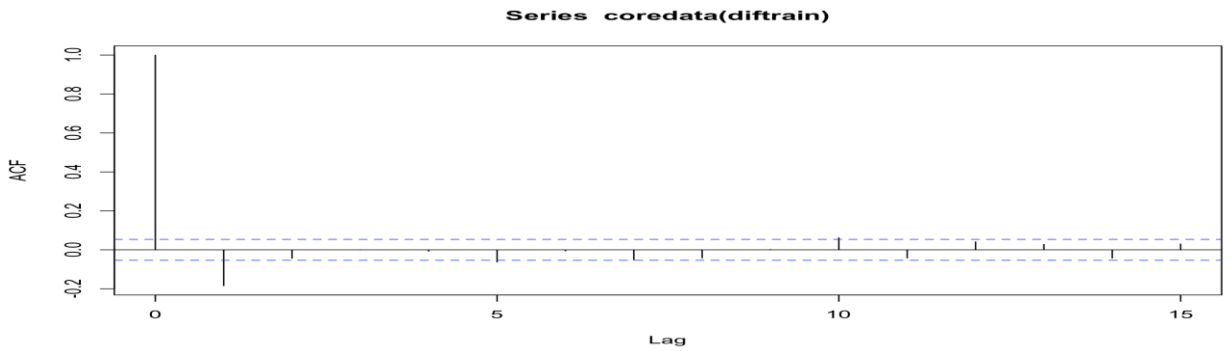
9. Perform Normality test

```
> normalTest(diftrain , method=c("jb"))

Title:
 Jarque - Bera Normalality Test

Test Results:
  STATISTIC:
    X-squared: 5.1114
  P VALUE:
    Asymptotic p Value: 0.07764

Description:
 Wed Apr 26 12:50:47 2017 by user:
```

we see that the p value is greater than 0.05 hence according to the test statistics the data is normally distributed.

10. Plot ACF graph

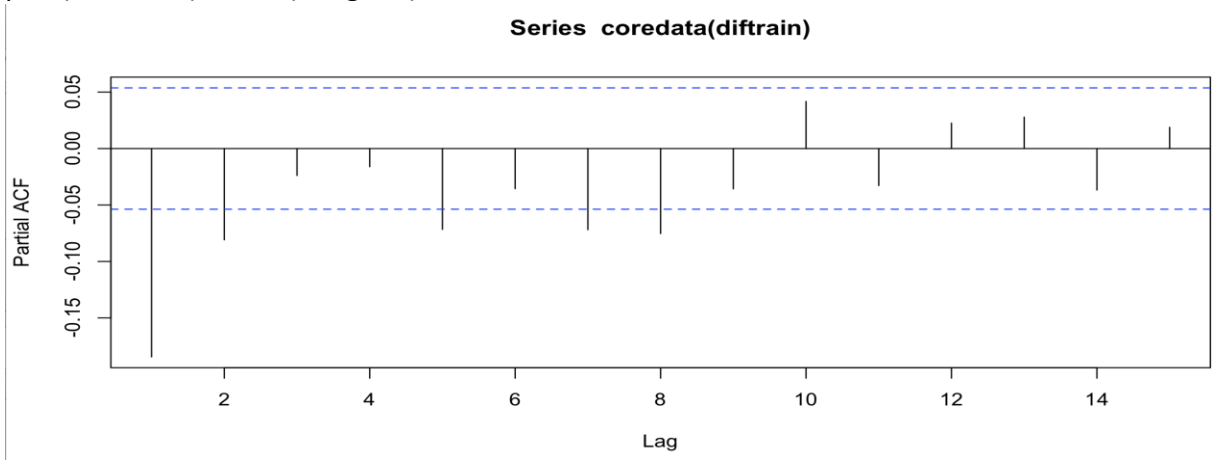
> #Plot ACF
> acf(coredata(diftrain) , lag=15)



plot PACF

> #plot PACF

pacf(coredata(diftrain) , lag=15)



11.Confirm of serial correlation by performing Ljung-Box test.

```
> Box.test(coredata(diftrain) ,lag=6 , type='Ljung')

    Box-Ljung test

data:  coredata(diftrain)
X-squared = 53.47, df = 6, p-value = 9.433e-10

> Box.test(coredata(diftrain) ,lag=12 , type='Ljung')

    Box-Ljung test

data:  coredata(diftrain)
X-squared = 69.269, df = 12, p-value = 4.387e-10
```

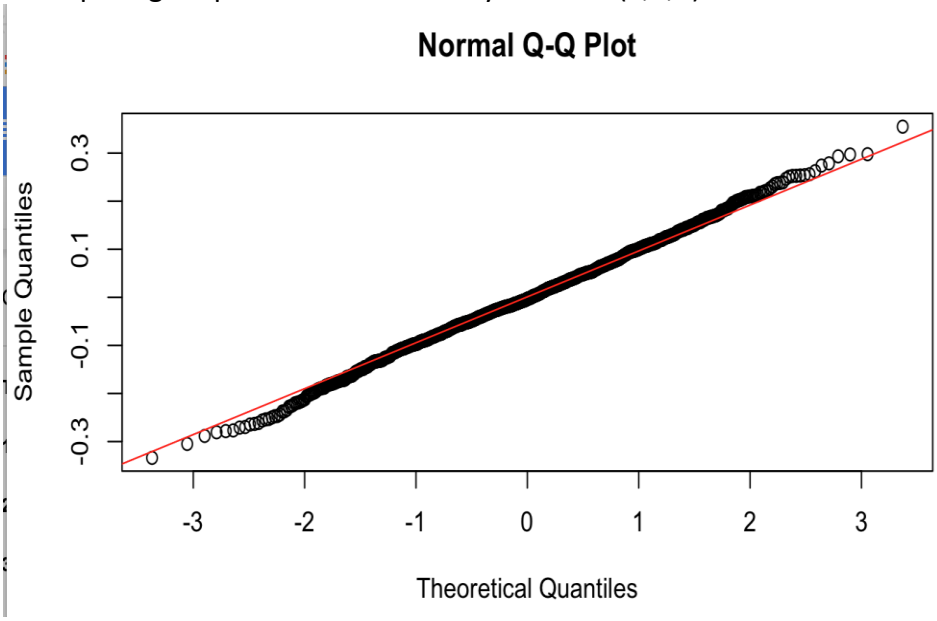
At lag 6 and 12 we have the p values to be less than 0.05 hence the series is serially correlated which means the data can be used for creating models for predicting the future.

12.Build the time series model:

- **AR model with p=2 and p=5**

```
> mAR1

Call:
arima(x = diftrain, order = c(2, 0, 0))

Coefficients:
          ar1      ar2  intercept
      -0.1996  -0.0807     0.0004
s.e.   0.0273   0.0274     0.0022

sigma^2 estimated as 0.01021:  log likelihood = 1162.26,  aic = -2316.52
```

Residual analysis:
#Computing QQplot for residual analysis for AR(2,0,0) model.



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for AR (2,0,0) model.

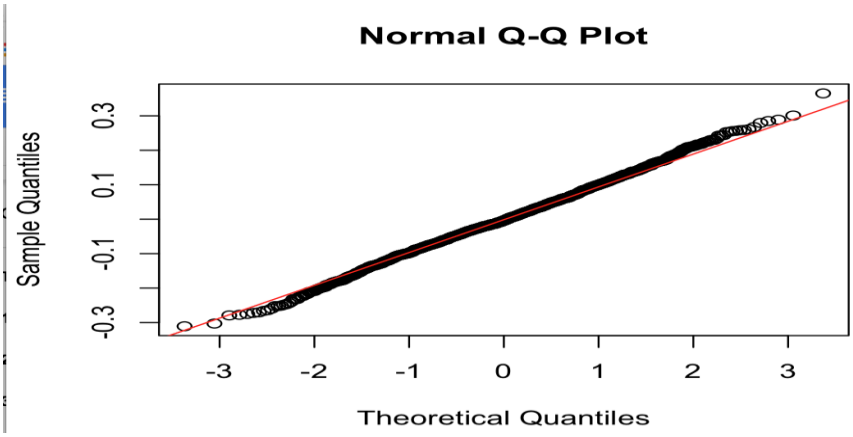
```
> #Computing Ljung Box test for residual analysis for AR model.
> Box.test(mAR1$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  mAR1$residuals
X-squared = 11.847, df = 6, p-value = 0.06547

> Box.test(mAR1$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  mAR1$residuals
X-squared = 30.17, df = 12, p-value = 0.002632
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

```
> mAR2=arima(diftrain,order=c(5,0,0))
> mAR2

Call:
arima(x = diftrain, order = c(5, 0, 0))

Coefficients:
          ar1      ar2      ar3      ar4      ar5  intercept
      -0.2032  -0.0887  -0.0325  -0.0313  -0.0733     0.0004
s.e.   0.0274   0.0280   0.0281   0.0281   0.0276     0.0019

sigma^2 estimated as 0.01015:  log likelihood = 1166.33,  aic = -2318.66
```

Residual analysis:

#Computing Q-Q plot for residual analysis for AR (5,0,0) model.



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say

that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for AR(5,0,0) model.

```
> #Computing Ljung Box test for residual analysis for AR model.
> Box.test(mAR2$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  mAR2$residuals
X-squared = 4.4892, df = 6, p-value = 0.6108

> Box.test(mAR2$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  mAR2$residuals
X-squared = 22.274, df = 12, p-value = 0.03457
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

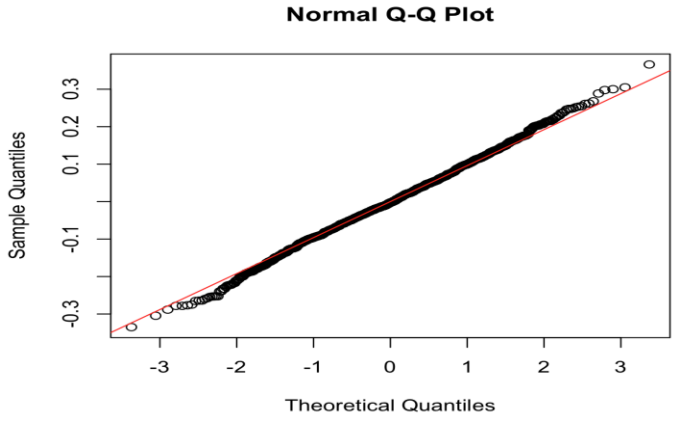
- **MA model**

MA model with q=1

```
> mMA1=arima(diftrain , order=c(0,0,1))
> mMA1

Call:
arima(x = diftrain, order = c(0, 0, 1))

Coefficients:
          ma1  intercept
      -0.2125     0.0004
s.e.   0.0284     0.0022

sigma^2 estimated as 0.01022:  log likelihood = 1161.39,  aic = -2316.78
```

Residual analysis:

#Computing QQplot for residual analysis for MA model.



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for MA model.

```
> #Computing Ljung Box test for residual analysis for MA model.
> Box.test(mMA1$residuals,lag=9,type='Ljung')

    Box-Ljung test

data:  mMA1$residuals
X-squared = 23.799, df = 9, p-value = 0.004631

> Box.test(mMA1$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  mMA1$residuals
X-squared = 31.393, df = 12, p-value = 0.001716
```

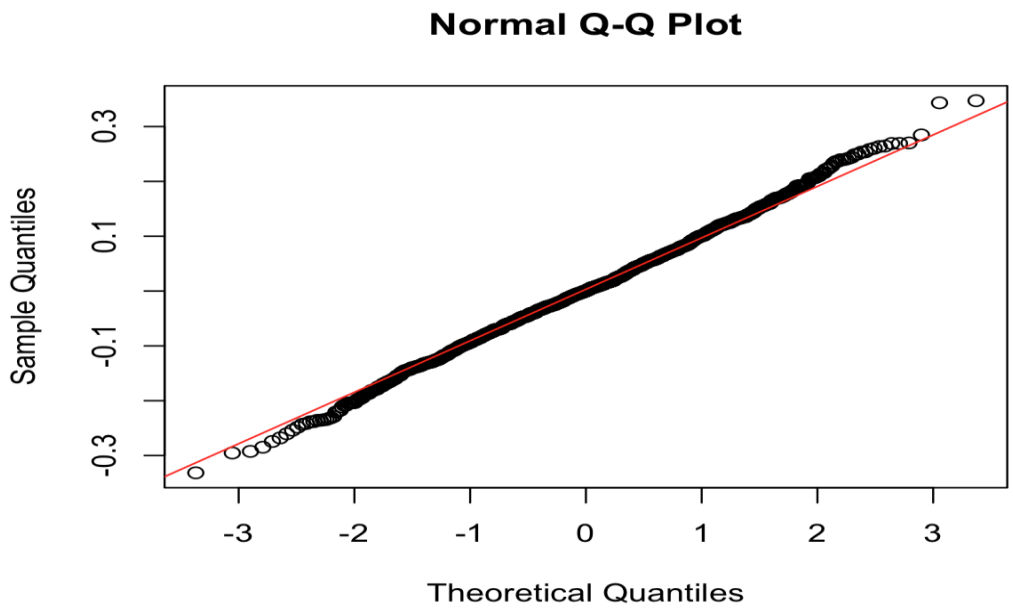
At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

- **ARIMA model using auto arima function**.

```
> monthlyARIMA = auto.arima(coredata(traindata1) ,allowdrift=FALSE)
>
> monthlyARIMA
Series: coredata(traindata1)
ARIMA(1,1,2)

Coefficients:
         ar1      ma1     ma2
      0.8934  -1.1335  0.1437
s.e.  0.0171   0.0329  0.0315

sigma^2 estimated as 0.009854:  log likelihood=1186.84
AIC=-2365.69   AICc=-2365.66   BIC=-2344.91
```

Residual Analysis:

#Computing Q-Q plot for residual analysis for ARIMA model.



From the above QQ plot, as we can see that the almost all the points lies on the line, we can say that the residual plots are normally distributed.

Computing Ljung Box test for residual analysis for ARIMA model.

```
> #Computing Ljung Box test for residual analysis for ARIMA model.
> Box.test(monthlyARIMA$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  monthlyARIMA$residuals
X-squared = 3.416, df = 6, p-value = 0.7551

> Box.test(monthlyARIMA$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  monthlyARIMA$residuals
X-squared = 17.935, df = 12, p-value = 0.1177
```

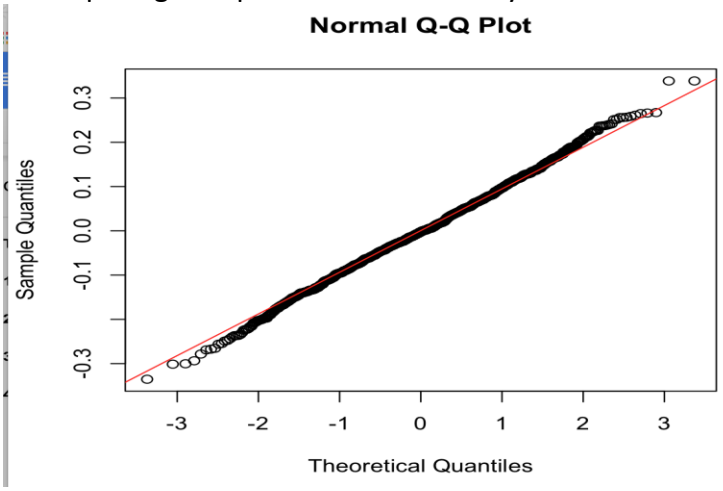
At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

- **ARMA model with p=2 and q=1**

```
> monthlyARMA= arima(diftrain, order=c(2,0,1),method='ML',include.mean=T)
> monthlyARMA

Call:
arima(x = diftrain, order = c(2, 0, 1), include.mean = T, method = "ML")

Coefficients:
         ar1     ar2      ma1  intercept
      0.753  0.1234  -0.9917      3e-04
s.e.  0.028  0.0278   0.0065      2e-04

sigma^2 estimated as 0.009822:  log likelihood = 1187.4,  aic = -2364.79
```

Residual Analysis:

#Computing Q-Q plot for residual analysis for ARMA model.



Computing Ljung Box test for residual analysis for ARMA model.

```
> #Computing Ljung Box test for residual analysis for ARMA model.
> Box.test(monthlyARMA$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  monthlyARMA$residuals
X-squared = 0.639, df = 6, p-value = 0.9957

> Box.test(monthlyARMA$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  monthlyARMA$residuals
X-squared = 16.038, df = 12, p-value = 0.1895
```

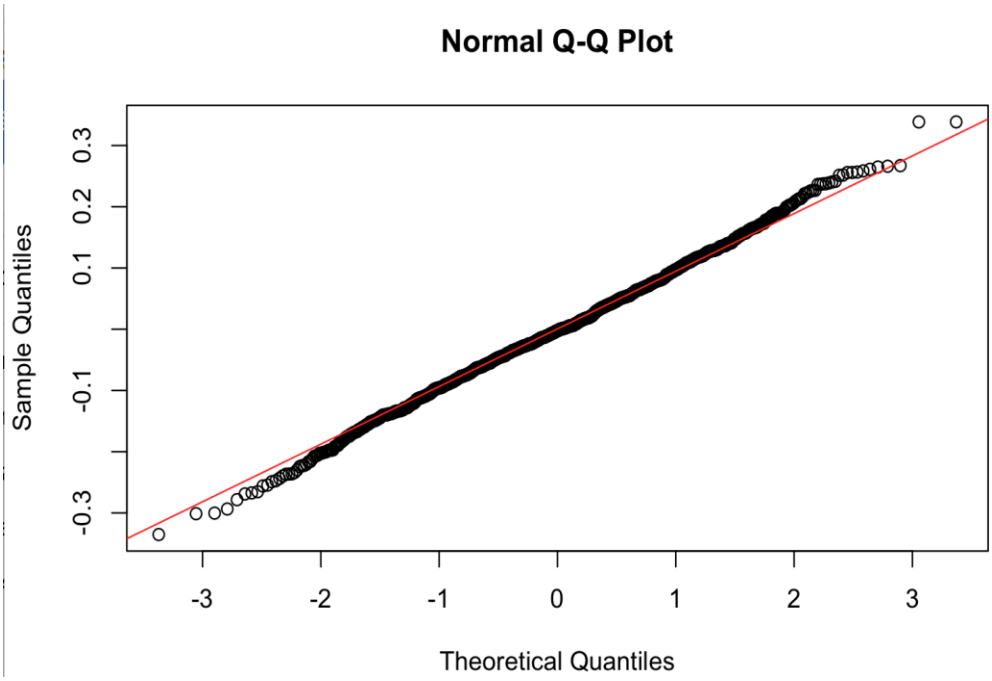
At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

- **ARMA model with p=5 and q=1**

```
> monthlyARMA1= arima(diftrain, order=c(5,0,1),method='ML',include.mean=T)
> monthlyARMA1

Call:
arima(x = diftrain, order = c(5, 0, 1), include.mean = T, method = "ML")

Coefficients:
         ar1     ar2     ar3      ar4      ar5      ma1  intercept
      0.7483  0.1041  0.0521  -0.0010  -0.0327  -0.9906      3e-04
s.e.  0.0281  0.0344  0.0344   0.0345   0.0280   0.0066      2e-04

sigma^2 estimated as 0.009797:  log likelihood = 1189.08,  aic = -2362.17
```

Residual Analysis:

#Computing QQplot for residual analysis for ARMA model.



Computing Ljung Box test for residual analysis for ARMA model.

```
> #Computing Ljung Box test for residual analysis for ARMA model.
> Box.test(monthlyARMA1$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  monthlyARMA1$residuals
X-squared = 0.639, df = 6, p-value = 0.9957

> Box.test(monthlyARMA1$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  monthlyARMA1$residuals
X-squared = 16.038, df = 12, p-value = 0.1895
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

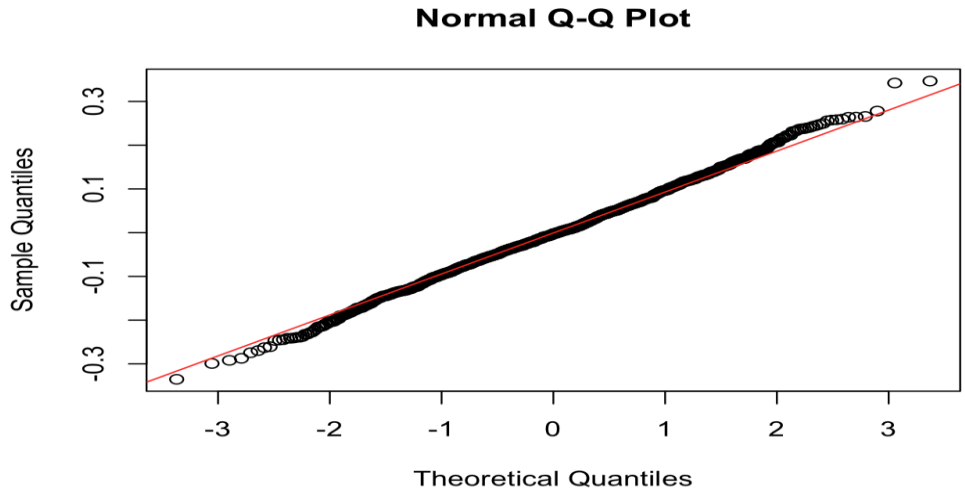
- **Building ARMA model with p=1 and q=2 by getting ECF value.**

```
> EACF(diftrain)
[1] "EACF table"
       [,1]   [,2]     [,3]     [,4]    [,5]     [,6]
[1,] -0.18 -0.044  0.00078 -0.00743 -0.0631 -0.0059
[2,] -0.37 -0.045 -0.00416  0.00035 -0.0619  0.0070
[3,] -0.27 -0.251  0.00247  0.00227 -0.0377  0.0220
[4,] -0.47  0.035 -0.02603  0.07788 -0.0265 -0.0173
[5,] -0.21 -0.301 -0.34463 -0.05858 -0.0757  0.0061
[6,] -0.40 -0.475 -0.24713  0.12873 -0.0045 -0.0303
[1] " "
[1] "Simplified EACF: 2 denotes significance"
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    0    0    0    2    0
[2,]    2    0    0    0    2    0
[3,]    2    2    0    0    0    0
[4,]    2    0    0    2    0    0
[5,]    2    2    2    2    2    0
[6,]    2    2    2    2    0    0
>
```

```
> monthlyARMA2= arima(diftrain, order=c(1,0,2),method='ML',include.mean=T)
> monthlyARMA2

Call:
arima(x = diftrain, order = c(1, 0, 2), include.mean = T, method = "ML")

Coefficients:
         ar1      ma1     ma2  intercept
      0.8989  -1.1412  0.1474      3e-04
s.e.  0.0190   0.0343  0.0320      2e-04

sigma^2 estimated as 0.009815:  log likelihood = 1187.81,  aic = -2365.61
```

Residual Analysis:

#Computing QQplot for residual analysis for ARMA model.



Computing Ljung Box test for residual analysis for ARMA model.

```
> #Computing Ljung Box test for residual analysis for ARMA model.
> Box.test(monthlyARMA2$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  monthlyARMA2$residuals
X-squared = 3.3575, df = 6, p-value = 0.7628

> Box.test(monthlyARMA2$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  monthlyARMA2$residuals
X-squared = 18.277, df = 12, p-value = 0.1075
```

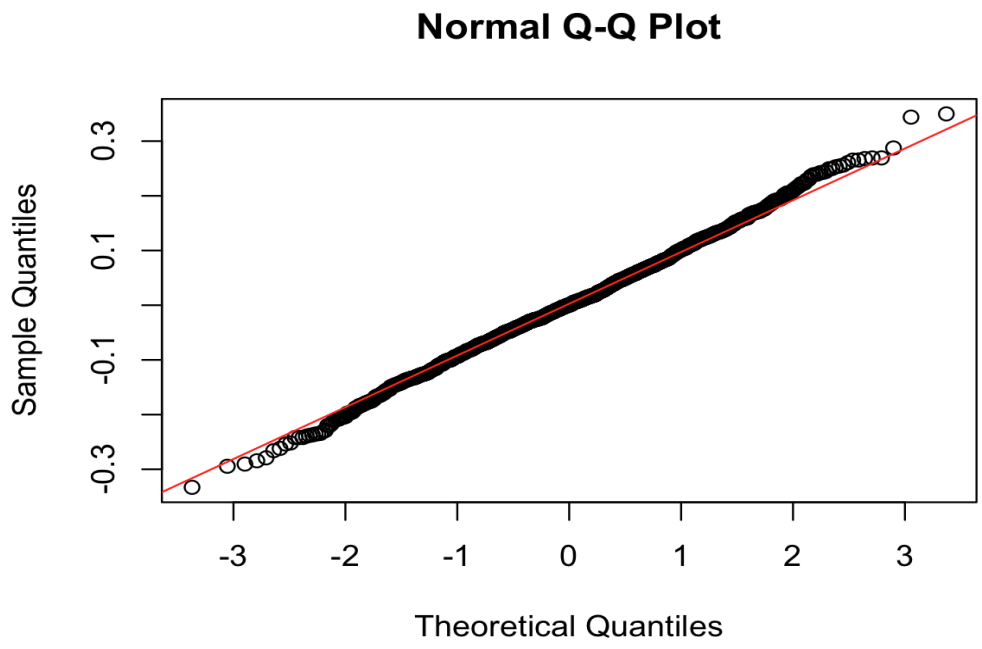
At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

- **Building ARMA model with auto.arima.**

```
> #Computing ARMA(p,q) model for Monthly data using auto.arima function
>  monthlyARMA3=auto.arima(coredata(diftrain), max.P=8, max.Q=8, ic="aic",allowdrift=FALSE)
>
> monthlyARMA3
Series: coredata(diftrain)
ARIMA(2,0,1) with zero mean

Coefficients:
         ar1     ar2      ma1
      0.7503  0.1213  -0.9876
s.e.  0.0278  0.0276   0.0053

sigma^2 estimated as 0.00986:  log likelihood=1186.48
AIC=-2364.97   AICc=-2364.94   BIC=-2344.19
```

Residual Analysis:

#Computing Q-Q plot for residual analysis for ARMA model.



Computing Ljung Box test for residual analysis for ARMA model.

```
> #Computing Ljung Box test for residual analysis for ARMA model.
> Box.test(monthlyARMA3$residuals,lag=6,type='Ljung')

    Box-Ljung test

data:  monthlyARMA3$residuals
X-squared = 4.3547, df = 6, p-value = 0.6288

> Box.test(monthlyARMA3$residuals,lag=12,type='Ljung')

    Box-Ljung test

data:  monthlyARMA3$residuals
X-squared = 18.762, df = 12, p-value = 0.09443
```

At 95% confidence level, we can see that the p-value is greater than 0.05, we conclude that residual is white noise, which meets the assumptions in residual analysis.

Checking the seasonality of the model.

Often time series possess a seasonal component that repeats every s observations. For monthly observations s = 12 (12 in 1 year), for quarterly observations s = 4 (4 in 1 year). In order to deal with seasonality, ARIMA processes have been generalized: SARIMA models have then been formulated.

seasonality in a time series model can be identified by checking the ACF plot for the data. notice If there are, any recurrent/seasonal effect in the ACF plot. Usually observe if there are only non-zero autocorrelations at lag 1, s-1, s, s+1

In our case, we can observe that in the ACF plot there is no seasonal trends observed.
the only non-zero variable that is present is on lag1.

Hence there is no need to apply SARIMA Model.

Best Model

As per using AIC value as the criteria, we can observe that the best model is ARIMA model that is got by auto.arima function.

The Best Model here being the below:

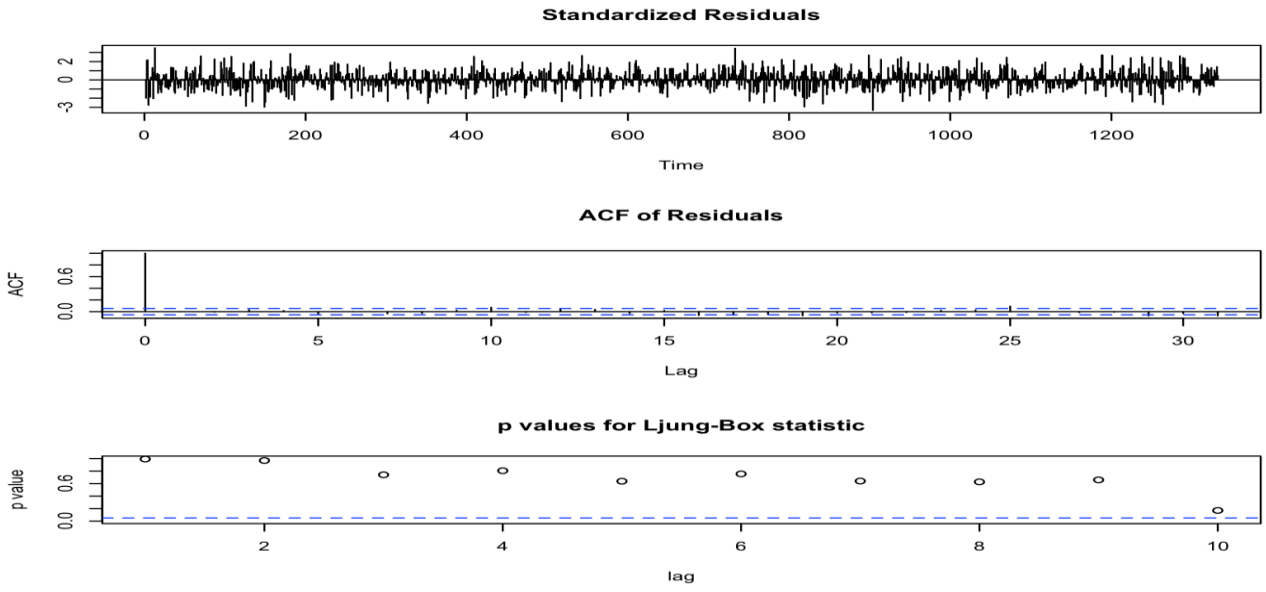
```
> monthlyARIMA = auto.arima(coredata(traindata1) ,allowdrift=FALSE)
>
> monthlyARIMA
Series: coredata(traindata1)
ARIMA(1,1,2)

Coefficients:
          ar1      ma1     ma2
      0.8934  -1.1335  0.1437
s.e.  0.0171   0.0329  0.0315

sigma^2 estimated as 0.009854:  log likelihood=1186.84
AIC=-2365.69   AICc=-2365.66   BIC=-2344.91
```

Residual Analysis:

-> tsdiag(monthlyARIMA)
res =monthlyARIMA$resid

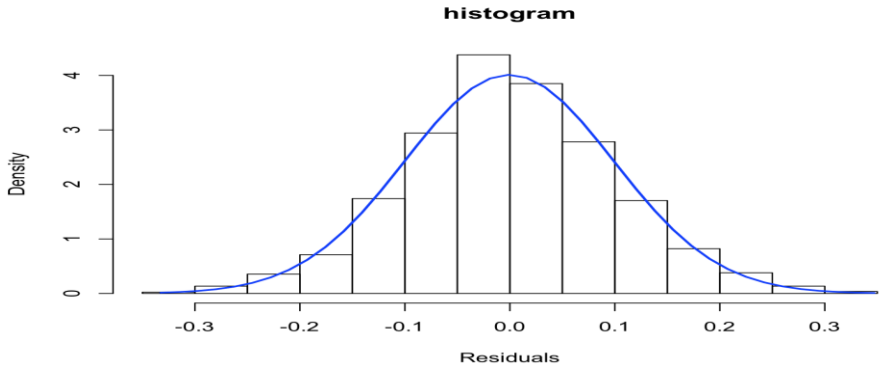
Plot the ACF model for the residues of the data, as we can observe that all the points in the plot are within the blue line which means that there is no autocorelation between the residual and we can state that the model seems to be statistically fitting the data and there is no violation of assumption of ARIMA model.

Ljung-Box statistics:
1) A small p-value is evidence that there is dependence.
So you want to see large p-values. But a large p-value
is not really evidence of independence -- merely a lack
of evidence of dependence.

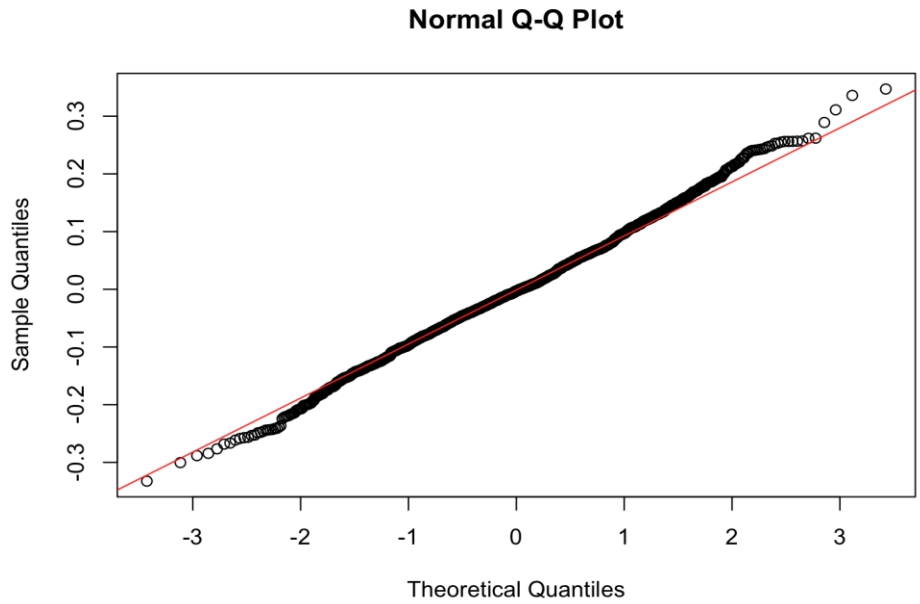


Analyse if the residuals are normally distributed

```
> hist(res, xlab="Residuals", prob=TRUE, main="histogram")
> xfit=seq(min(res), max(res), length=40)
> yfit=dnorm(xfit, mean=mean(res), sd=sd(res))
> lines(xfit, yfit, col="blue", lwd=2)
```



Plot the normality graph:



With respect to the graph we can say that the residuals are normally distributed.

Ljung Box test for residuals:

```
> Box.test(coredata(res),lag=10, type='Ljung')

    Box-Ljung test

data:  coredata(res)
X-squared = 17.326, df = 10, p-value = 0.06746
```

```
> Box.test(coredata(res) ,lag=12 ,type='Ljung')

     Box-Ljung test

data:  coredata(res)
X-squared = 17.342, df = 12, p-value = 0.1372
```

In general, the Ljung Box test is defined as:
Ho = The model does not exhibit lack of fit.
Ha = The model exhibit lack of fit.

Hence with if the p value for the test is > 0.05 we can't reject null hypothesis.

Comparing the model based on MAE values:

Predictions:
predicting value for AR models:

> #predicting the value for AR model
> ar_predict=predict(mAR1, n.ahead=314, se.fit=T)
> ar_predict

```
> #predicting the value for AR model
> ar_predict=predict(mAR1, n.ahead=314, se.fit=T)
> ar_predict
$pred
               Jan           Feb           Mar           Apr           May           Jun           Jul
1991 -0.0078094517  0.0023897352  0.0006216855  0.0001516476  0.0003881294  0.0003788528  0.0003616232
1992  0.0003659796  0.0003659793  0.0003659801  0.0003659799  0.0003659799  0.0003659799  0.0003659799
1993  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799
1994  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799
1995  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799  0.0003659799
```

```
>  #predicting the value for AR model
> ar_predict1=predict(mAR2, n.ahead=314, se.fit=T)
> ar_predict1
$pred
               Jan           Feb           Mar           Apr           May           Jun           Jul
1991 -1.887962e-02 -9.740804e-04 -5.255465e-03 -5.904328e-03  3.141661e-03  1.994864e-03  2.676967e-04
1992  3.417299e-04  3.640596e-04  3.959547e-04  3.671463e-04  3.627613e-04  3.687828e-04  3.662490e-04
1993  3.670651e-04  3.668745e-04  3.669328e-04  3.669664e-04  3.669408e-04  3.669461e-04  3.669583e-04
1994  3.669509e-04  3.669509e-04  3.669507e-04  3.669508e-04  3.669509e-04  3.669508e-04  3.669508e-04
```

```
> #predicting the value for MA model
> ar_predict2=predict(mMA1, n.ahead=314, se.fit=T)
> ar_predict2
$pred
               Jan           Feb           Mar           Apr           May           Jun           Jul
1991 -0.0043139922  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708
1992  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708
1993  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708
1994  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708
1995  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708  0.0003701708
```

```
> #predicting the value for Auto.arima model
> ar_predict3=predict(monthlyARIMA, n.ahead=314, se.fit=T)
> ar_predict3
$pred
Time Series:
Start = 1333
End = 1646
Frequency = 1
  [1] 0.29685315 0.27253693 0.25081191 0.23140194 0.21406034 0.19856670 0.18472409 0.17235658 0.16130697
 [10] 0.15143484 0.14261469 0.13473444 0.12769393 0.12140367 0.11578371 0.11076262 0.10627660 0.10226861
 [19] 0.09868772 0.09548842 0.09263004 0.09007626 0.08779461 0.08575610 0.08393482 0.08230762 0.08085381
 [28] 0.07955493 0.07839446 0.07735765 0.07643132 0.07560371 0.07486428 0.07420366 0.07361343 0.07308609
 [37] 0.07261495 0.07219402 0.07181794 0.07148194 0.07118174 0.07091353 0.07067390 0.07045981 0.07026853
 [46] 0.07009764 0.06994495 0.06980854 0.06968666 0.06957777 0.06948049 0.06939357 0.06931591 0.06924653
 [55] 0.06918454 0.06912916 0.06907968 0.06903547 0.06899597 0.06896068 0.06892916 0.06890099 0.06887582
>  #predicting the value for ARMA model
> ar_predict4=predict(monthlyARMA, n.ahead=314, se.fit=T)
> ar_predict4
$pred
             Jan           Feb           Mar           Apr           May           Jun           Jul
1991 -3.260910e-02 -1.906096e-02 -1.985668e-02 -2.221857e-02 -1.946430e-02 -1.678320e-02 -1.505624e-02
1992 -6.784318e-03 -5.916740e-03 -5.154400e-03 -4.484457e-03 -3.896432e-03 -3.380257e-03 -2.927044e-03
1993 -1.160183e-03 -9.780354e-04 -8.181283e-04 -6.777460e-04 -5.545045e-04 -4.463110e-04 -3.513280e-04
1994  1.895842e-05  5.713153e-05  9.064366e-05  1.200639e-04  1.458919e-04  1.685663e-04  1.884721e-04
1995  2.660739e-04  2.740740e-04  2.810972e-04  2.872629e-04  2.926757e-04  2.974276e-04  3.015993e-04
1996  3.178625e-04  3.195391e-04  3.210110e-04  3.223032e-04  3.234375e-04  3.244334e-04  3.253077e-04

> #predicting the value for ARMA model
> ar_predict5=predict(monthlyARMA1, n.ahead=314, se.fit=T)
> ar_predict5
$pred
             Jan           Feb           Mar           Apr           May           Jun           Jul
1991 -3.260910e-02 -1.906096e-02 -1.985668e-02 -2.221857e-02 -1.946430e-02 -1.678320e-02 -1.505624e-02
1992 -6.784318e-03 -5.916740e-03 -5.154400e-03 -4.484457e-03 -3.896432e-03 -3.380257e-03 -2.927044e-03
1993 -1.160183e-03 -9.780354e-04 -8.181283e-04 -6.777460e-04 -5.545045e-04 -4.463110e-04 -3.513280e-04
1994  1.895842e-05  5.713153e-05  9.064366e-05  1.200639e-04  1.458919e-04  1.685663e-04  1.884721e-04
1995  2.660739e-04  2.740740e-04  2.810972e-04  2.872629e-04  2.926757e-04  2.974276e-04  3.015993e-04

> ar_predict6=predict(monthlyARMA2, n.ahead=314, se.fit=T)
> ar_predict6
$pred
             Jan           Feb           Mar           Apr           May           Jun           Jul
1991 -3.400825e-02 -2.395169e-02 -2.149569e-02 -1.928807e-02 -1.730371e-02 -1.552002e-02 -1.391672e-02
1992 -7.182600e-03 -6.422457e-03 -5.739187e-03 -5.125016e-03 -4.572956e-03 -4.076726e-03 -3.630679e-03
1993 -1.757213e-03 -1.545737e-03 -1.355648e-03 -1.184783e-03 -1.031197e-03 -8.931432e-04 -7.690508e-04
1994 -2.478432e-04 -1.890096e-04 -1.361259e-04 -8.859027e-05 -4.586191e-05 -7.454643e-06  2.706851e-05

> #predicting the value for ARMA model
> ar_predict7=predict(monthlyARMA3, n.ahead=314, se.fit=T)
> ar_predict7
$pred
Time Series:
Start = 1332
End = 1645
Frequency = 1
  [1] -3.280028e-02 -2.515509e-02 -2.285172e-02 -2.019638e-02 -1.792472e-02 -1.589827e-02 -1.410232e-02
  [8] -1.250906e-02 -1.109583e-02 -9.842258e-03 -8.730311e-03 -7.743989e-03 -6.869098e-03 -6.093049e-03
 [15] -5.404676e-03 -4.794073e-03 -4.252454e-03 -3.772025e-03 -3.345874e-03 -2.967868e-03 -2.632568e-03
 [22] -2.335149e-03 -2.071331e-03 -1.837319e-03 -1.629744e-03 -1.445621e-03 -1.282299e-03 -1.137429e-03
```

Converting the predicted difference value to the prediction value of the original data, and calculating MAE values.

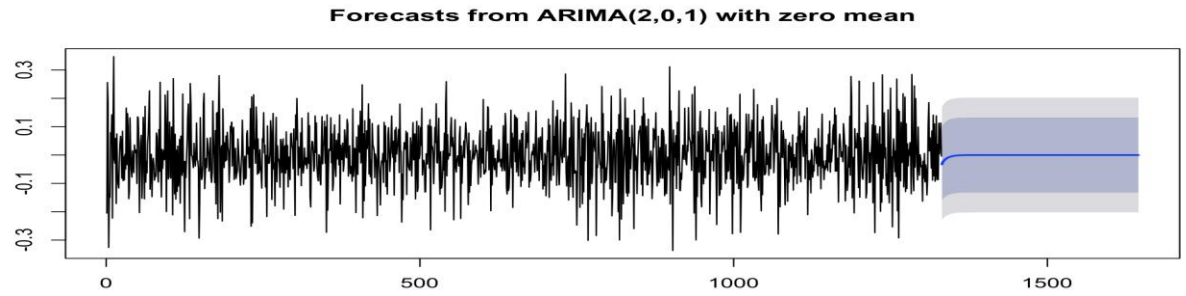
Loading the predicted and differenced values file.
>predictionval=read.table("/Users/arpithajayarama/Documents/predictedval.csv",header=T,sep=','
,')
> predictionval

```
> predictionval=read.table("/Users/arpithajayarama/Documents/predictedval.csv",header=T,sep=',')
> predictionval
    year anomaly.of.temperature..k. Predicted.AR      Dif.AR predicted.AR1      Dif.AR1 predicted.MA
1   1991                  0.257911 -0.007809452  0.32349055 -0.0188796150  0.312420385 -0.004313992
2   1991                  0.302717  0.002389735  0.26030074 -0.0009740800  0.256936920  0.000370171
3   1991                  0.237145  0.000621686  0.30333869 -0.0052554650  0.297461535  0.000370171
4   1991                  0.243740  0.000151648  0.23729665 -0.0059043280  0.231240672  0.000370171
```

```
> par=predictionval[,4]
> mae_ar=sum(abs(observedval-par)/nrow(testdata))
> mae_ar
[1] 0.08201817
>
> par1=predictionval[,6]
> mae_ar1=sum(abs(observedval-par1)/nrow(testdata))
> mae_ar1
[1] 0.0819964
>
> pma=predictionval[,8]
> mae_ma=sum(abs(observedval-pma)/nrow(testdata))
> mae_ma
[1] 0.08203444
>
> parima=predictionval[,10]
> mae_arima=sum(abs(observedval-parima)/nrow(testdata))
> mae_arima
[1] 0.1084042
>
> parma=predictionval[,12]
> mae_arma=sum(abs(observedval-parma)/nrow(testdata))
> mae_arma
[1] 0.08198631
>
> parma1=predictionval[,14]
> mae_arma1=sum(abs(observedval-parma1)/nrow(testdata))
> mae_arma1
[1] 0.08198631
>
> parma2=predictionval[,16]
> mae_arma2=sum(abs(observedval-parma2)/nrow(testdata))
> mae_arma2
[1] 0.08198551
>
> parma3=predictionval[,18]
> mae_arma3=sum(abs(observedval-parma3)/nrow(testdata))
> mae_arma3
[1] 0.08197441
>
```

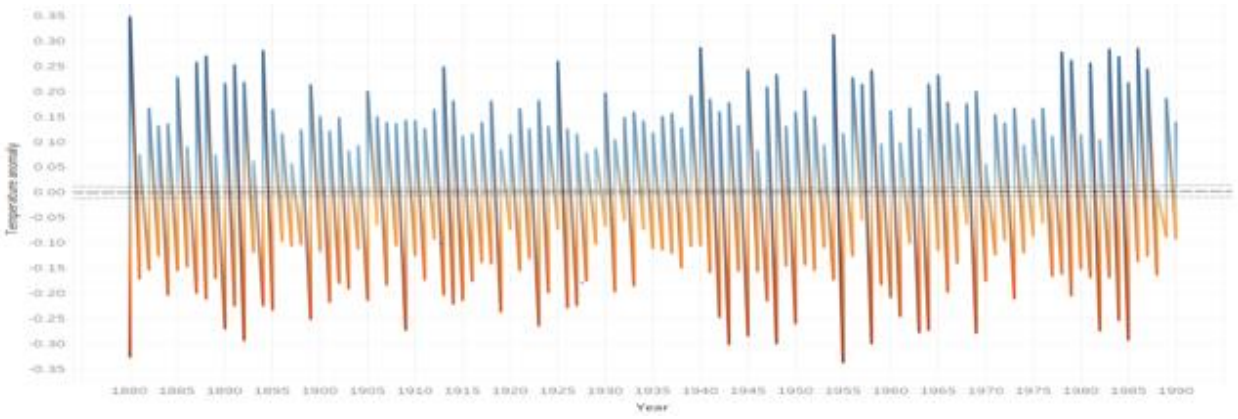
With respect the above MAE value the monthlyARMA3 has the least value and hence it is considered to be the best model.

When plotted the forecast for the model, the below plot is produced, which shows very less variation in the data.



Hence, we investigated even further regarding the trends in the data set.

Plot the time series graph to analyze the trends in the data set



As we can see from the below plot there's almost no trend in the data. When there's no trend, we go for exponential smoothing techniques. below is the model produced by applying exponential smoothing.

```
> model
ETS(A,N,A)

Call:
 ets(y = diftrain)

  Smoothing parameters:
    alpha = 1e-04
    gamma = 0.0296

  Initial states:
    l = 0.003
    s=0.0427 0.0486 0.0223 0.0344 -0.0021 -0.0206
           -0.0015 -0.0333 -0.0069 -0.0294 0.0241 -0.0782

  sigma:  0.1008

     AIC     AICc      BIC
3497.267 3497.632 3575.172
```

For any time period t, the smoothed value St is found by computing:
St=αyt−1+(1−α)St−10<α≤1t≥3.
This is the basic equation of exponential smoothing and the constant or parameter is called the smoothing constant.
Basic equation can be expanded by first substituting in for St-1 the basic equation to obtain
St=αyt−1+(1−α)[αyt−2+(1−α)St−2]αyt−1+α(1−α)yt−2+(1−α)2St−2.]

5. Evaluations and Results

5.1. Evaluation Methods

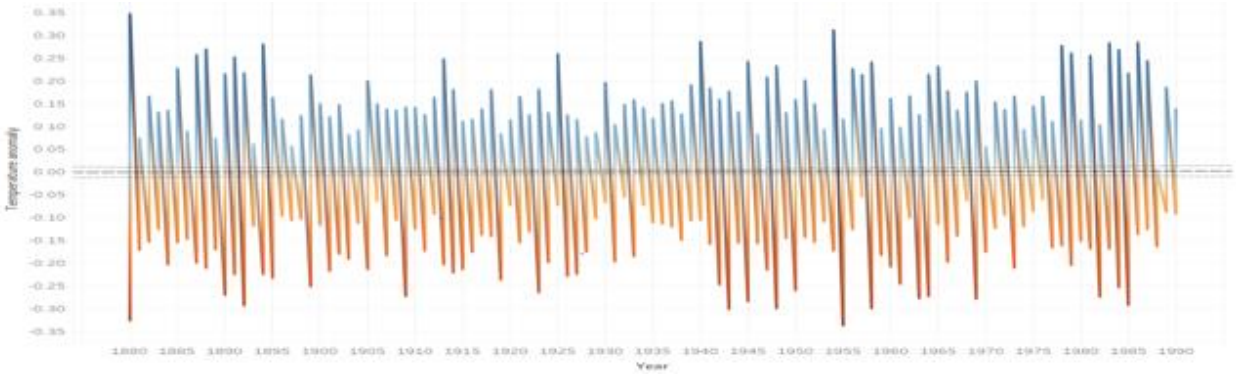
Evaluation is basically a research which might be about deciding how compelling the model is, which parts of it are functioning admirably and which require modifying, or whether a few parameters react to specific strategies or conditions uniquely in contrast to others. In the event that the models are dependable, we need to give the evaluation a structure that will reveal us what we need to know. We move ahead with hold-out evaluation as the dataset is very large enough. In this, we divide the dataset into "training set" and "test set" i.e. 80% and 20% and perform evaluation operation. We are going to build the model based on “training set” and use “test set” for evaluation. We will be taking into consideration one of the metrics like AIC, BIC or CP mallow statistics.

5.2. Results and Findings

Created different models like AR(p), MA(q), ARMA (p ,q)and ARIMA(p,0,q).

We have forecasted future values using these models. Finally, after analyzing the forecasted values, we have decided to go forward with AR model for yearly data analysis and for monthly data, exponential smoothing model. The reason behind this is, forecasted values using this model are very close to the actual values in the dataset. The best model based on yearly data is used for forecasting and predictions.

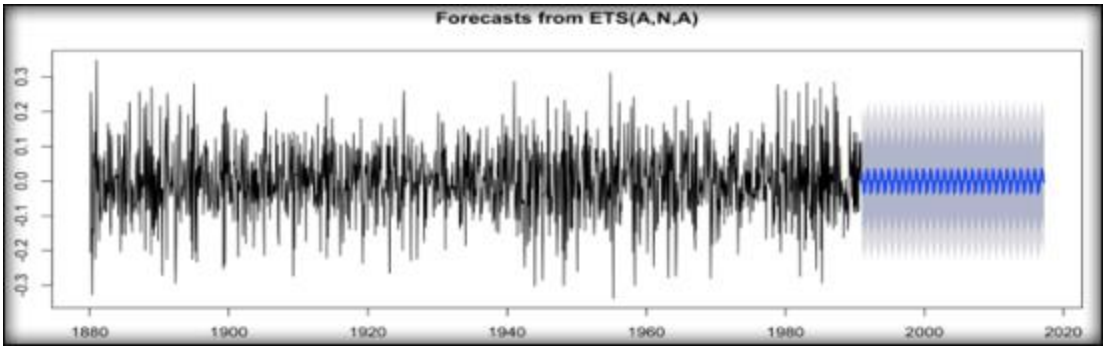
Findings based on monthly data analysis:



As we can see from the above plot there’s almost no trend in the data. When there’s no trend, we go for exponential smoothing techniques.

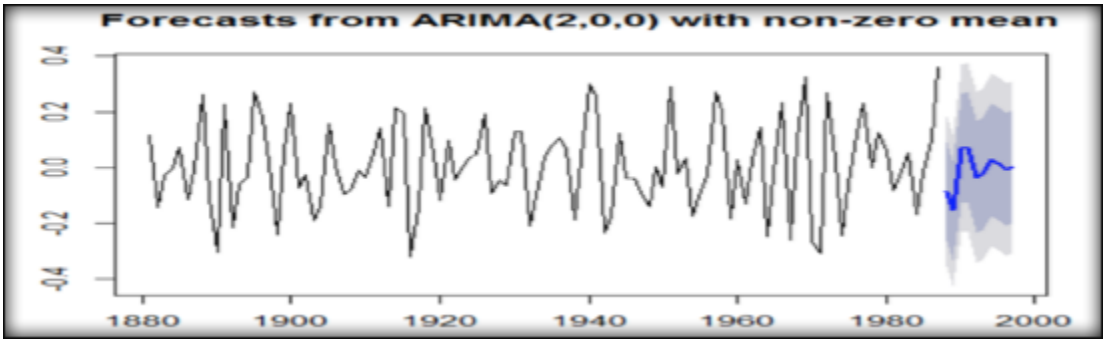
Model using exponential smoothing techniques based on monthly data which is used for forecasting is shown below:

Forecasted plot based on monthly data analysis



As we have applied exponential smoothing, we have got additive forecast with spikes over the entire time span. The plot above shows the forecasted values are over 95% confidence region, the shaded region represents that itself. On performing additional analysis, we can get exact future values which will lie anywhere in this confidence region.

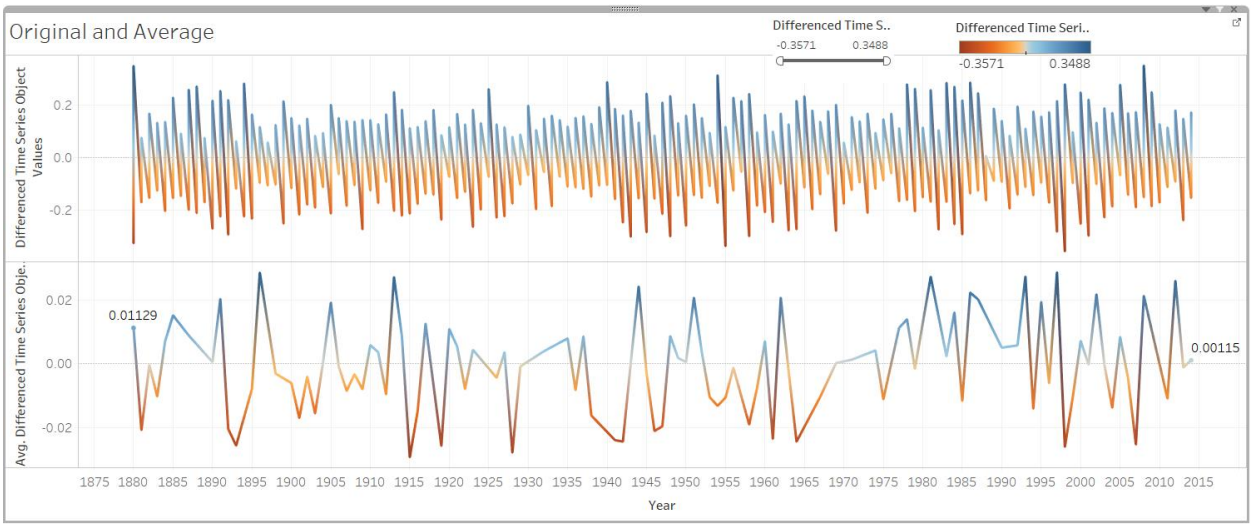
Forecasted plot based on yearly data analysis



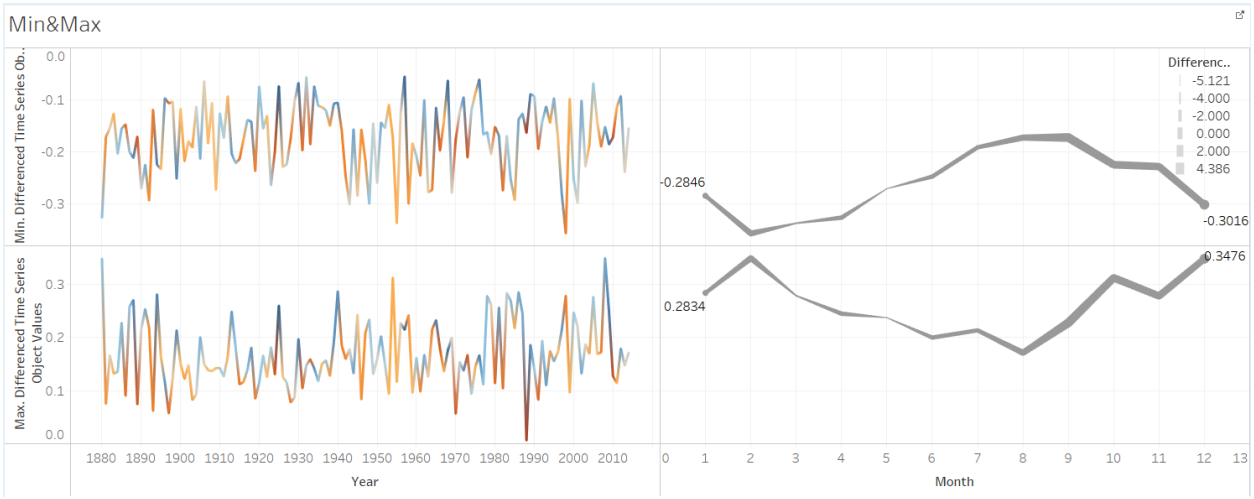
As we can see from the forecast plot above, there is slight downfall in temperature anomaly leaded by increase which gradually comes to stable level over the time span. The plot above shows the forecasted values are over 95% confidence region, the shaded region represents that itself. On performing additional analysis, we can get exact future values which will lie anywhere in the confidence region.

5.3. Data Visualization with Tableau

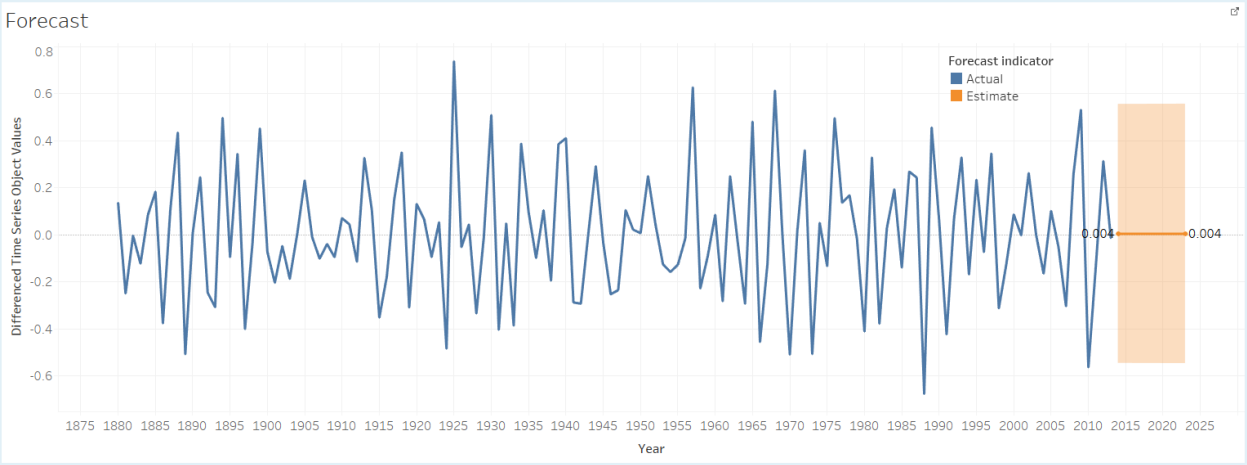
The plot below is the time series representation of the dataset from 1880-2017. The one below it, has the average of all the temperature anomalies in respective years plotted in time series form.



The plots on the left shows minimum and maximum anomalies in temperature on yearly basis from 1880-2017. The plots on the right shows minimum and maximum anomalies in temperature on monthly basis from 1880-2017.



The plot below shows time series representation for temperature anomalies from 1880-2016. Also it gives the forecasted plot which is quite linear. The forecasted region under 95% confidence interval. So for more accuracy, the future values can lie anywhere in that region.



6. Conclusions and Future Work

6.1. Conclusions

The temperature anomaly time series fitted by exponential smoothing model for the monthly dataset and AR(p) model for yearly dataset can be used for forecasting temperature anomaly based on the data of past 137 years and also for estimating the missing values if there are any. Based on these models which fits the best, the temperature anomaly will slightly rise from that of the reference period 1880-2016. As we advance in time, the uncertainty about this predictions grows and there can be variations in temperature anomaly.

6.2. Limitations

There are limitations in this dataset which holds us back from getting to the most or almost near to accurate results. Some of the factors which are missing which plays an influential role are latitude, altitude, ocean currents, prevailing winds, distance from sea. A dataset which considers all these factors can give a better prediction on temperature anomaly.

6.3. Potential Improvements or Future Work

Potential improvements or Future Work to this can involve collecting data for all the factors which plays an important role towards temperature of an area. Using a dataset of such a type can help scientists or decision makers to establish better strategies to build systems which stands as an arming tool against upcoming weather changes. It can be a savior for animal life, food cycle, biodiversity, human life against abrupt climatic conditions.