End to End Time series project for seasonal and non-seasonal datasets

Author : Deepshika Reddy AG

CWID: 10473464

TABLE OF CONTENTS

1. Introduction and Motivation

The paper presented here explains 2 kinds of datasets , one being seasonal and other being non-seasonal datasets. Time-series is a branch of data science that deals with univariate data with respect to date time. It is very useful for data that are particularly serially correlated. Handling time series data is quite challenging because it is quite difficult to understand the trend it may produce. For some datasets the trend can be totally random , while for others it can be seasonal or cyclic in nature . Time series analysis can be performed only if the dataset is stationary in nature. Throughout the paper , we first preprocess the dataset , make the dataset univariate and make the date time as index . Next steps involve plotting of time series plot,acf , pacf and eacf graphs which helps in identifying the model to be used. Also we need to check if the dataset is stationary using Dicky-Fuller test. If it proves to be stationary we can directly analyze the bars in acf and eacf and come up with AR or MA model . If its not stationary we can apply techniques like diffrencing,transforming and detrending to convert them into stationary . Further we apply ARIMA models and perform parameter estimation using AIC,BIC and so on. Further in the paper , various concepts pertaining to residual analysis is performed like ACF plot,histogram,qqplot,Shapiro-wilk test and Ljung-box plot . Prediction is based on forecasting on the original dataset for the future values and see how time series perform.

2. Data source identification

Seasonal dataset has been derived from Kaggle which is a temperature change dataset for different months. Kaggle is a good source for collecting any kind of dataset as there is clear description of the fields and the dataset is readily available in csv file.

Non-Seasonal dataset has been collected from Fred official website which has large collection of time series datasets for various categories to chose from . The univariate dataset is readily available for time series analysis with clear description . It also has a time series plot already plotted so we can chose the dataset with a certain trend . I feel it's a easy and great learning to capture datasets from the fred website which has both financial and non-financial dataset.

3. Data set Description

1. Seasonal Dataset:

The FAOSTAT temperature change dataset contains the mean temperature change by country along with their annual updates . The time duration of the dataset goes from 1961-2019. The dataset has statistics available for monthly, seasonal and annual mean temperatures. For the analysis purpose we have converted the columns with years to a single column and have filtered out only temperature change data's' and have ignored global warming and climate change respectively. By the problem statement it is known that temperature change can vary for every month of each year which makes it the seasonal part and can be clearly seen in the time series plot as well.

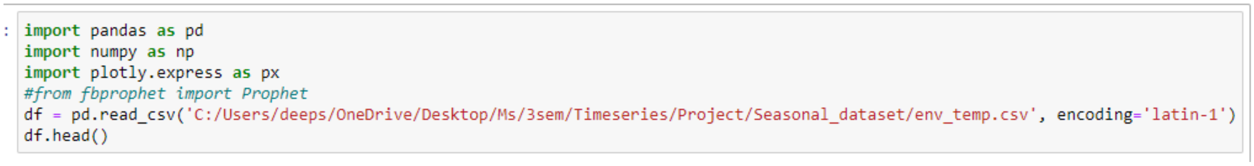
2. Non-Seasonal Dataset:

The dataset is a unemployment rate dataset for over 20 years . It has been collected from a household survey for population and formulated the .csv file with date and the percentage of unemployment rate over the years. The data has been collected from the source , "US Bureau of Labor Statistics" which has been present in the fred website. The dataset talks about the employment situation in USA , which is a monthly data and is seasonally adjusted.

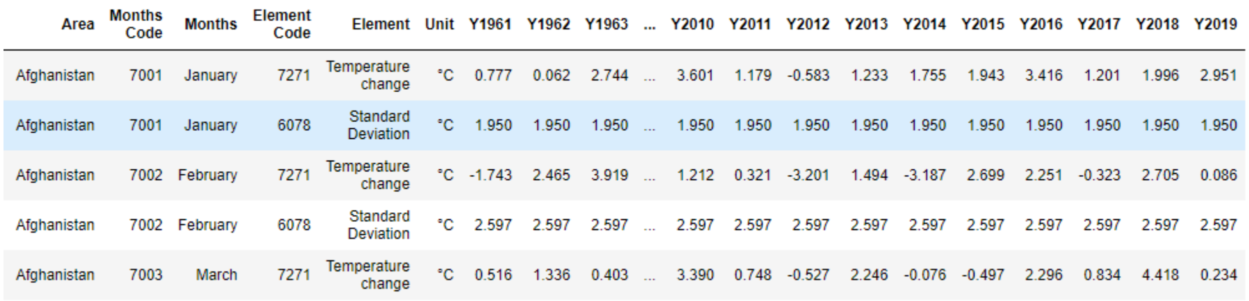
4. Arima Seasonal Time series modeling

4.1 Data Preprocessing:

End to End Time series project for seasonal and non-seasonal datasets

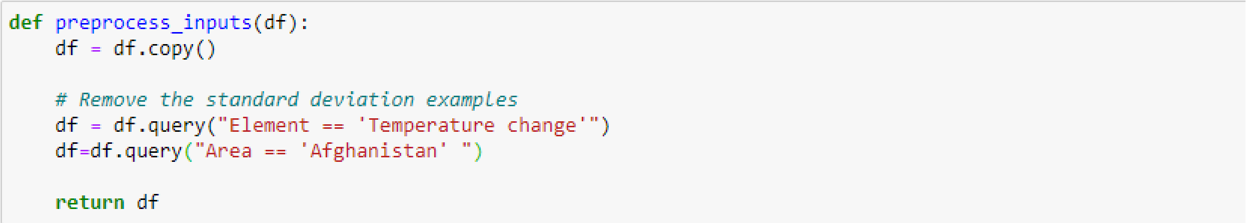


```
import pandas as pd
import numpy as np
import plotly.express as px
#from fbprophet import Prophet
df = pd.read_csv('C:/Users/deeps/OneDrive/Desktop/Ms/3sem/Timeseries/Project/Seasonal_dataset/env_temp.csv', encoding='latin-1')
df.head()
```



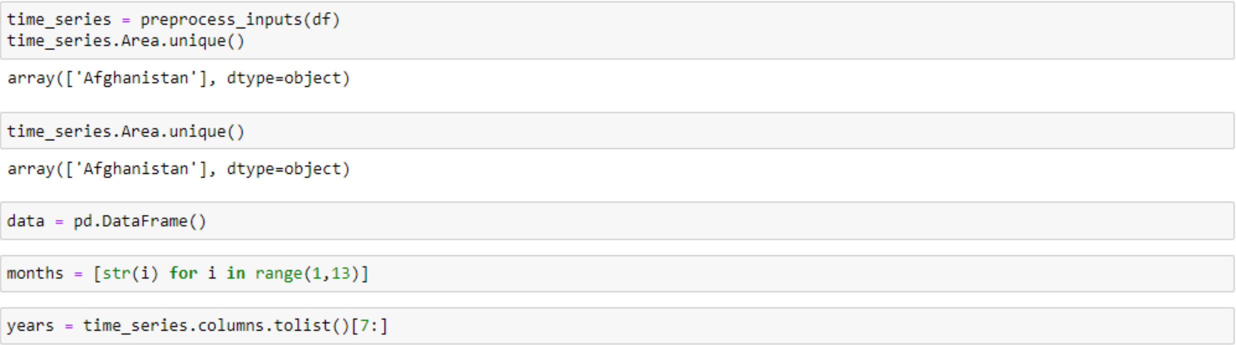
Area	Months Code	Months	Element Code	Element	Unit	Y1961	Y1962	Y1963	...	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2018	Y2019
Afghanistan	7001	January	7271	Temperature change	°C	0.777	0.062	2.744	...	3.601	1.179	-0.583	1.233	1.755	1.943	3.416	1.201	1.996	2.951
Afghanistan	7001	January	6078	Standard Deviation	°C	1.950	1.950	1.950	...	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950	1.950
Afghanistan	7002	February	7271	Temperature change	°C	-1.743	2.465	3.919	...	1.212	0.321	-3.201	1.494	-3.187	2.699	2.251	-0.323	2.705	0.086
Afghanistan	7002	February	6078	Standard Deviation	°C	2.597	2.597	2.597	...	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597	2.597
Afghanistan	7003	March	7271	Temperature change	°C	0.516	1.336	0.403	...	3.390	0.748	-0.527	2.246	-0.076	-0.497	2.296	0.834	4.418	0.234

Shows the sample data from the csv file read. It has been seen that it contains values of temperature change , standard deviation and so on values from the years 1961 to 2019. For the scope of the project we are doing only time series analysis of temperature change so other elements data needs to be removed. Also we can see that the data is distributed in column wise for years which has to be row wise as the data frame will have date as index and temperature change value as the field. Time series is a univariate analysis so that format of the data requires changes which will be discussed in the further section.



```
def preprocess_inputs(df):
    df = df.copy()

    # Remove the standard deviation examples
    df = df.query("Element == 'Temperature change'")
    df=df.query("Area == 'Afghanistan' ")

    return df
```

Shows the preprocessing logic to filter only the required data.



```
time_series = preprocess_inputs(df)
time_series.Area.unique()
```

```
array(['Afghanistan'], dtype=object)
```

```
time_series.Area.unique()
```

```
array(['Afghanistan'], dtype=object)
```

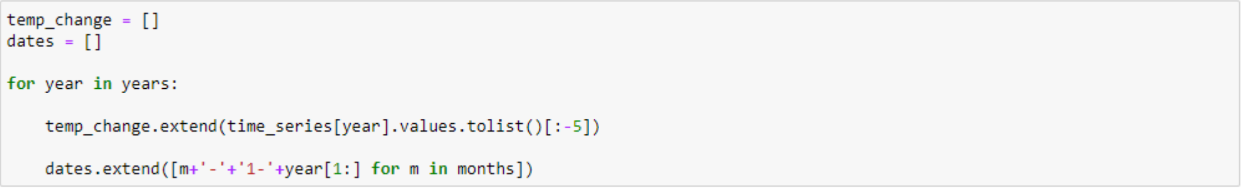
```
data = pd.DataFrame()
```

```
months = [str(i) for i in range(1,13)]
```

```
years = time_series.columns.tolist()[7:]
```

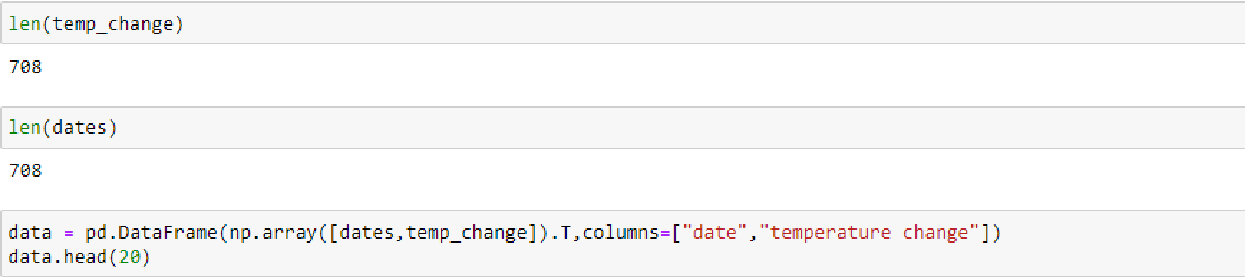
We calculate the data range for year by slicing the data frame column and storing in a year variable.

End to End Time series project for seasonal and non-seasonal datasets



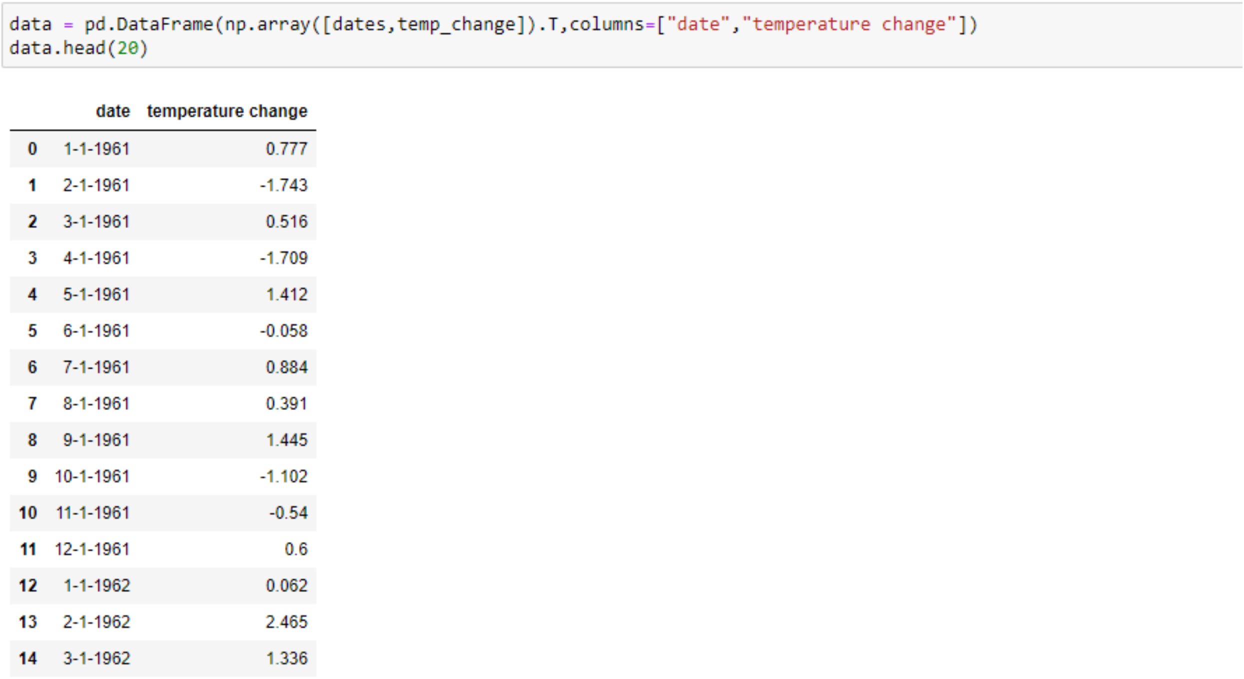
```
temp_change = []
dates = []

for year in years:

    temp_change.extend(time_series[year].values.tolist()[:-5])

    dates.extend([m+'-'+'1-'+year[1:] for m in months])
```

This is the main logic for forming the time series dataset. Here every year value we iterate and store the value in tem_change list. Also the date is formed by calculating the month from the month field and the year from the year field and convert the same into a single data point.



```
len(temp_change)

708

len(dates)

708

data = pd.DataFrame(np.array([dates,temp_change]).T,columns=["date","temperature change"])
data.head(20)
```

Shows the number of values found in the data frame. This is to make sure date and temp change values are in same dimension.

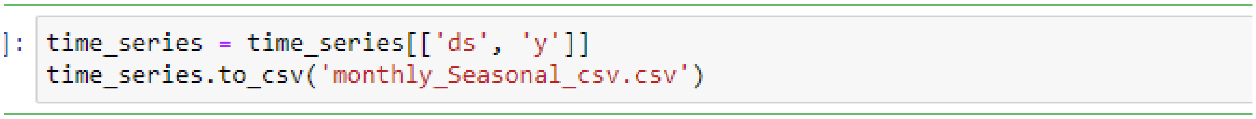


```
data = pd.DataFrame(np.array([dates,temp_change]).T,columns=["date","temperature change"])
data.head(20)
```

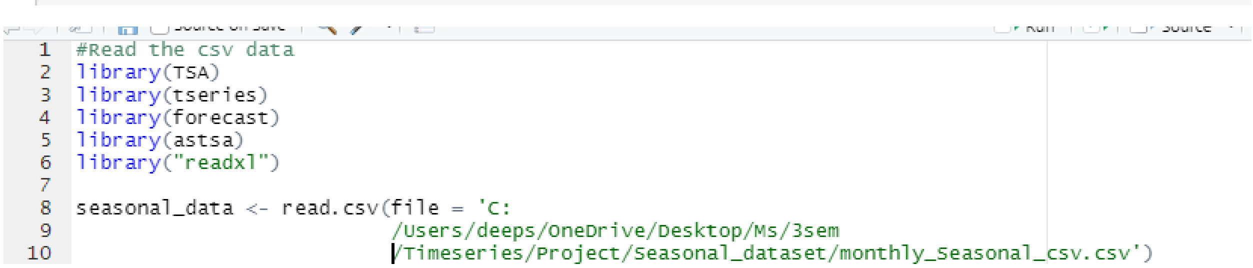
	date	temperature change
0	1-1-1961	0.777
1	2-1-1961	-1.743
2	3-1-1961	0.516
3	4-1-1961	-1.709
4	5-1-1961	1.412
5	6-1-1961	-0.058
6	7-1-1961	0.884
7	8-1-1961	0.391
8	9-1-1961	1.445
9	10-1-1961	-1.102
10	11-1-1961	-0.54
11	12-1-1961	0.6
12	1-1-1962	0.062
13	2-1-1962	2.465
14	3-1-1962	1.336

Sample time series data.

End to End Time series project for seasonal and non-seasonal datasets

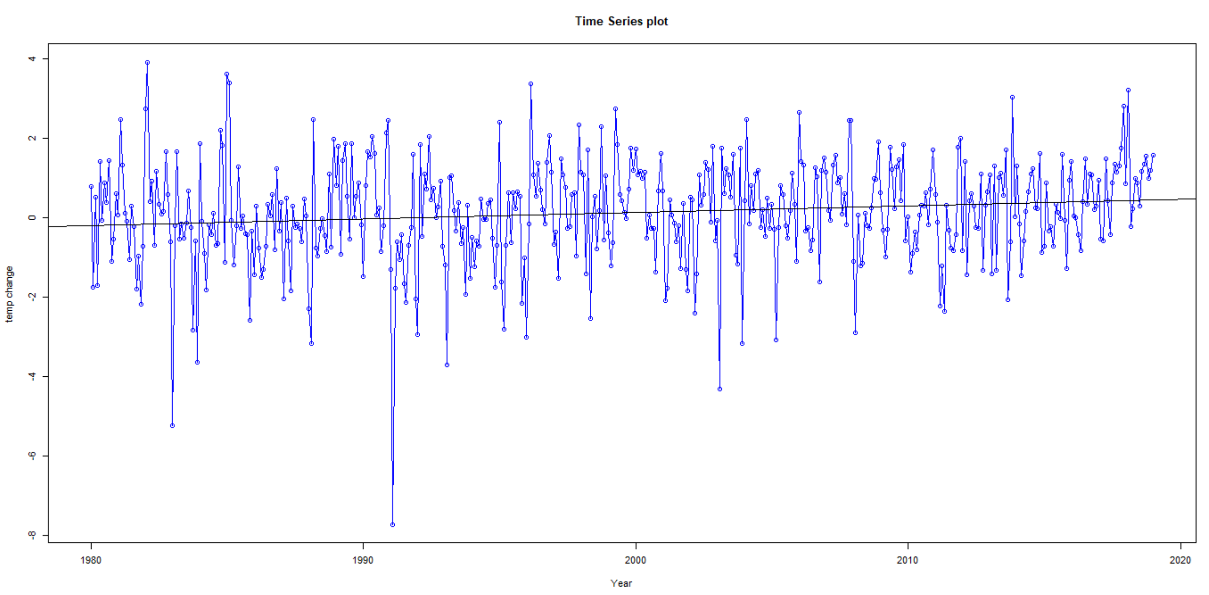


```
time_series = time_series[['ds', 'y']]
time_series.to_csv('monthly_Seasonal_csv.csv')
```

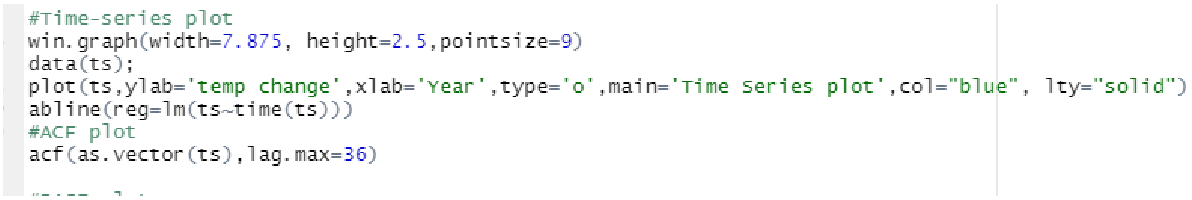


```
#Read the csv data
library(TSA)
library(tseries)
library(forecast)
library(astsa)
library("readxl")

seasonal_data <- read.csv(file = 'C:
                          /Users/deeps/OneDrive/Desktop/MS/3sem
                          /Timeseries/Project/Seasonal_dataset/monthly_Seasonal_csv.csv')
```

Sample csv data reading in R.

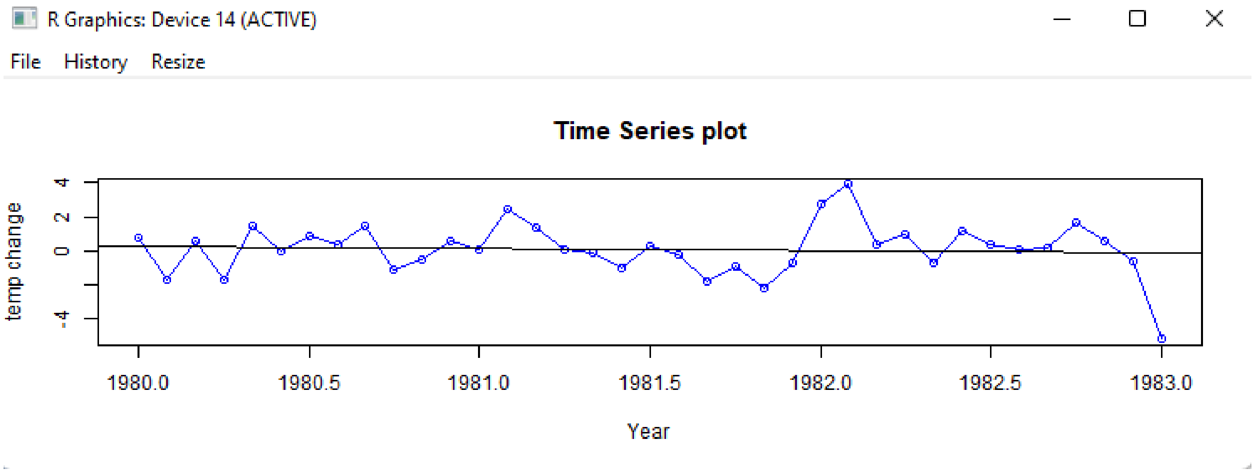


Time series plot for the given dataset. We can see that the data is stationary has there is no much variation or trend found(like upward or downward trend). But only the DF test can confirm stationary . Now if we analyze a sample data we can see the seasonally trends .

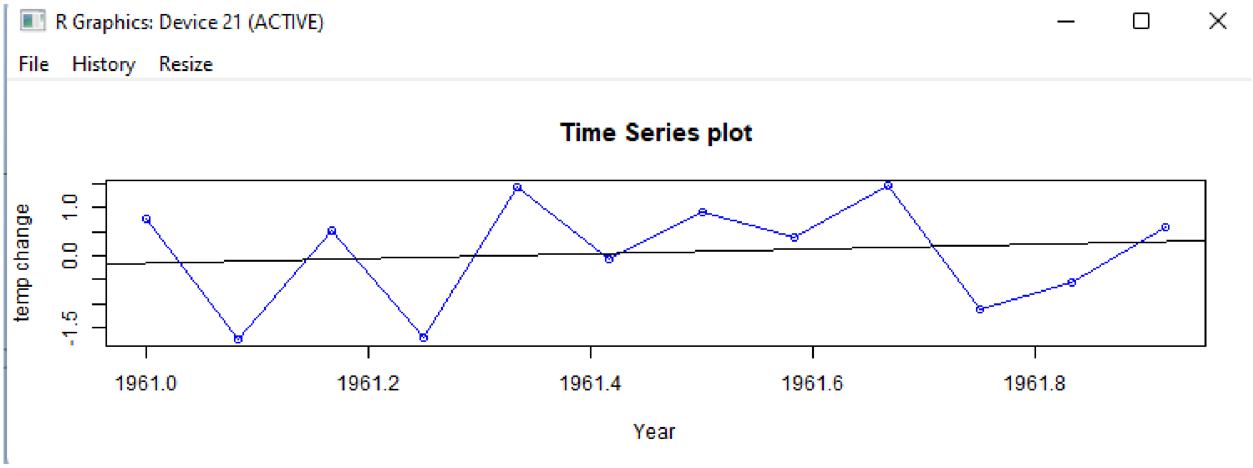


```
#Time-series plot
win.graph(width=7.875, height=2.5,pointsize=9)
data(ts);
plot(ts,ylab='temp change',xlab='Year',type='o',main='Time Series plot',col="blue", lty="solid")
abline(reg=lm(ts~time(ts)))
#ACF plot
acf(as.vector(ts),lag.max=36)
```

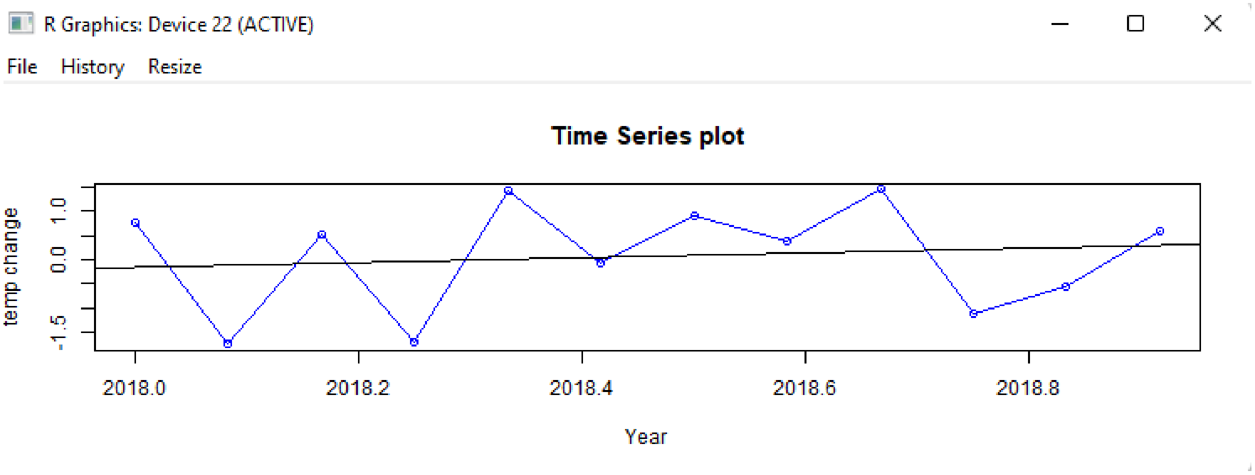
End to End Time series project for seasonal and non-seasonal datasets



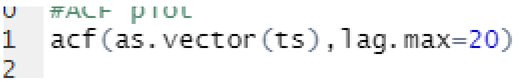
Find the sample data between 1980 Jan to 1983 Jan where the seasonality can be found. We can see that there is an increase in temperature change from Jan to February slightly and little higher drop to March . In the mid of the year the temperature change is almost constant there is no much difference and in the year 1983 we can see sudden drop in temperature at January when compared to previous years.



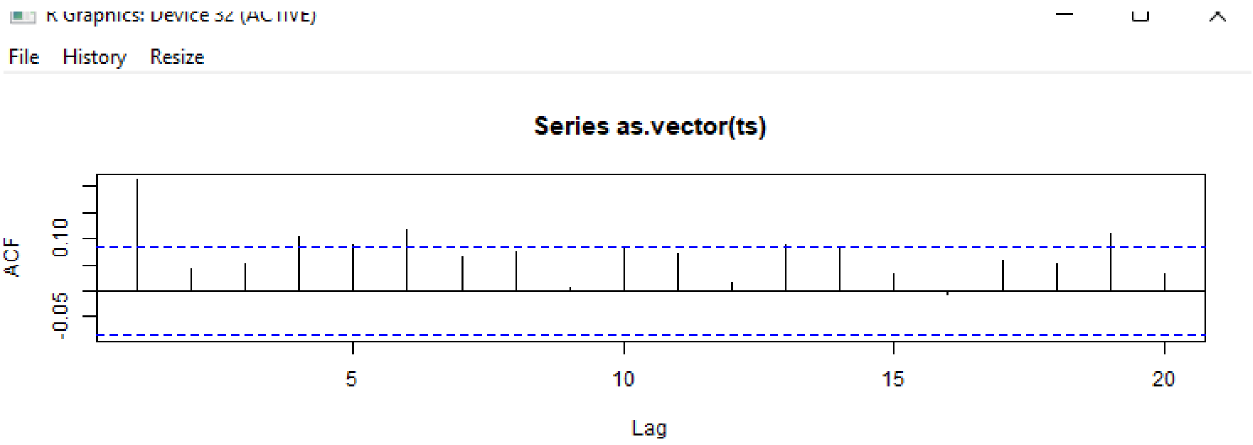
Shows the time series plot for the year 1961, where there is hike at March and July for the temperature change. Same pattern has been found even for the year 2018 so we can clearly find the seasonality from the time series plot.



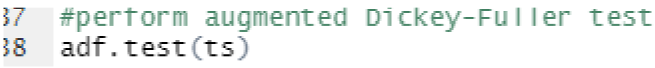
4.2 ACF plot:



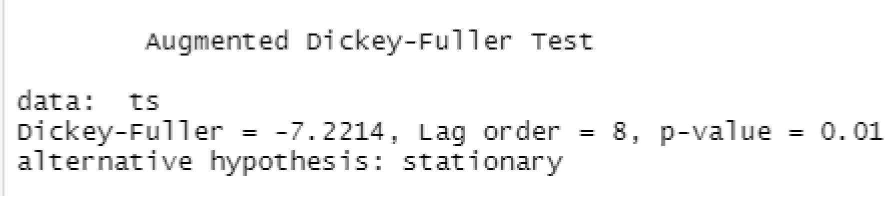
```
#ACF plot
acf(as.vector(ts),lag.max=20)
```



ACF plot signifies that there are 2 points outside the confidence interval. We can see that ACF curve shows the seasonality trends . The correlation has been found for multiple points beyond the confidence interval. We can see strong correlation at points 1,3,6,19 and so on. However the DF test , confirms that the time series is stationary as the p values is close to zero and also less than 0.05.Pins in the graph indicate MA(2) and 3 and 6 indicate the seasonal parts for MA(2).



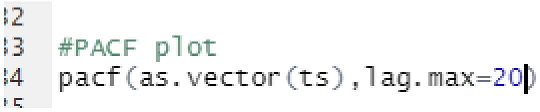
```
#perform augmented Dickey-Fuller test
adf.test(ts)
```



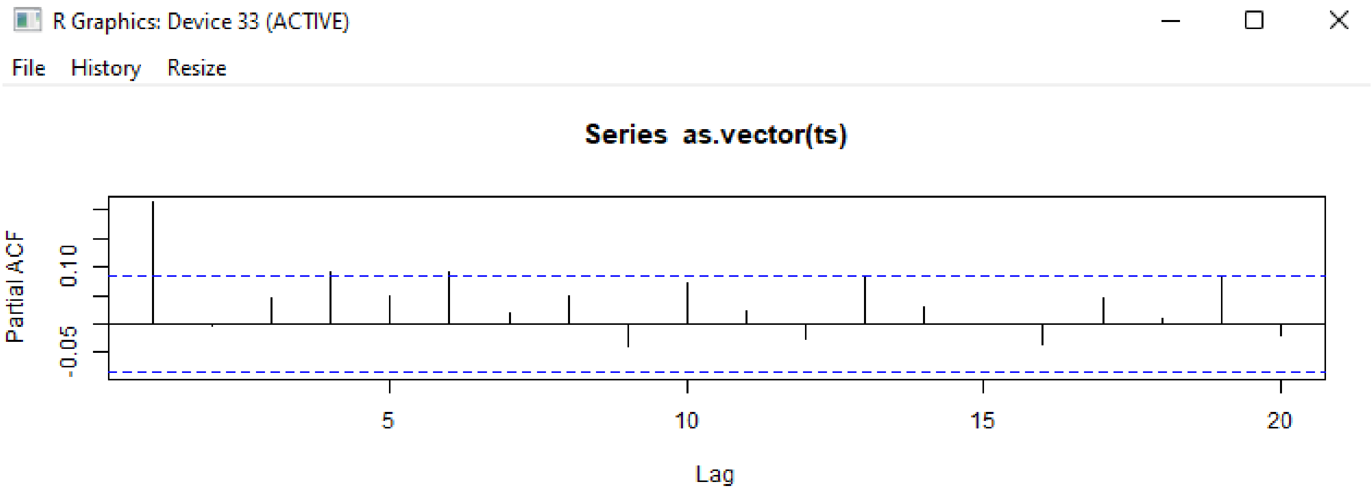
```

        Augmented Dickey-Fuller Test

data:  ts
Dickey-Fuller = -7.2214, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary
```

We can find the dickey-fuller test confirming the p values , which confirms the stationary.

4.3 PACF plot:



```
#PACF plot
pacf(as.vector(ts),lag.max=20)
```



We can see that only 1,3,6 pins are significant so we can use AR(3) part for non seasonal , however in the 6 can be used to find the seasonal part which is again AR(1) .

4.4 Modeling:

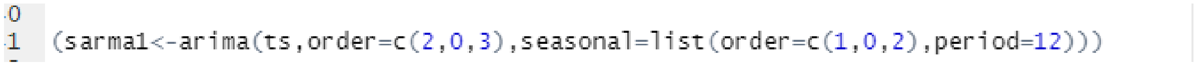
Based on the above analysis we can form the SARMA model as,

SARMA(,0,)X(,0,)

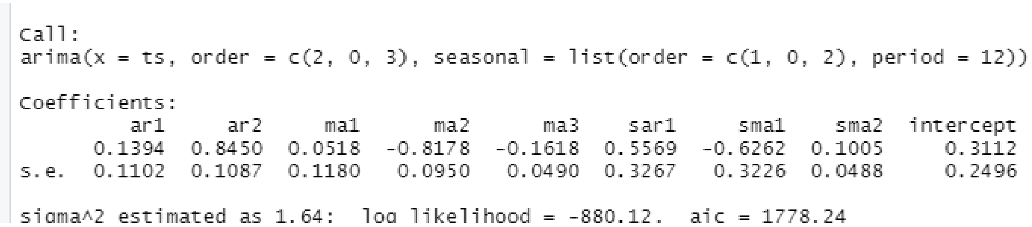
Has no differentiation has been done we can mark it as zero. First part of multiplication is the Non-seasonal part with first parameter as PACF and second as ACF . Similarly it’s the same format for Seasonal part as well in SARMA model.

From the above ACF,PACF analysis we can formulate the below models:

1.SARMA(2,0,3)X(1,0,2)



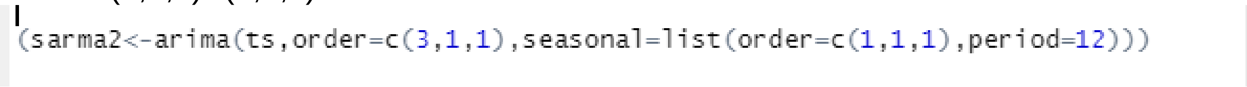
```
(sarma1<-arima(ts,order=c(2,0,3),seasonal=list(order=c(1,0,2),period=12)))
```



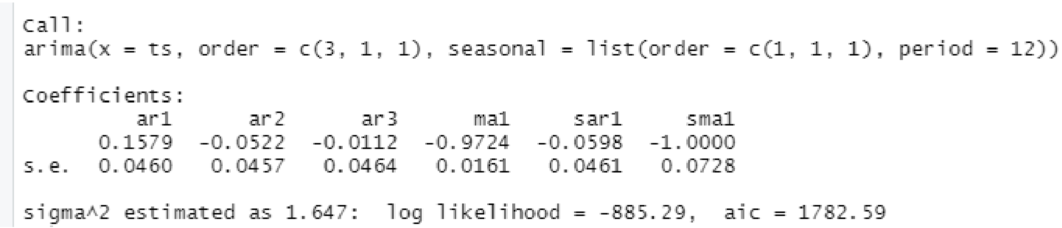
```
Call:
arima(x = ts, order = c(2, 0, 3), seasonal = list(order = c(1, 0, 2), period = 12))

Coefficients:
         ar1     ar2     ma1      ma2      ma3    sar1     sma1    sma2  intercept
      0.1394  0.8450  0.0518  -0.8178  -0.1618  0.5569  -0.6262  0.1005     0.3112
s.e.  0.1102  0.1087  0.1180   0.0950   0.0490  0.3267   0.3226  0.0488     0.2496

sigma^2 estimated as 1.64:  log likelihood = -880.12,  aic = 1778.24
```

We can see that , the AIC value is quite large , but we need to relatively check the AIC value with another model and then decide which is better . AIC value will only tell us measure of relative quality of statistical model for a given set of data .

2. SARMA(3,0,1)X(1,0,1)



```
(sarma2<-arima(ts,order=c(3,1,1),seasonal=list(order=c(1,1,1),period=12)))
```



```
Call:
arima(x = ts, order = c(3, 1, 1), seasonal = list(order = c(1, 1, 1), period = 12))

Coefficients:
         ar1      ar2      ar3      ma1     sar1     sma1
      0.1579  -0.0522  -0.0112  -0.9724  -0.0598  -1.0000
s.e.  0.0460   0.0457   0.0464   0.0161   0.0461   0.0728

sigma^2 estimated as 1.647:  log likelihood = -885.29,  aic = 1782.59
```

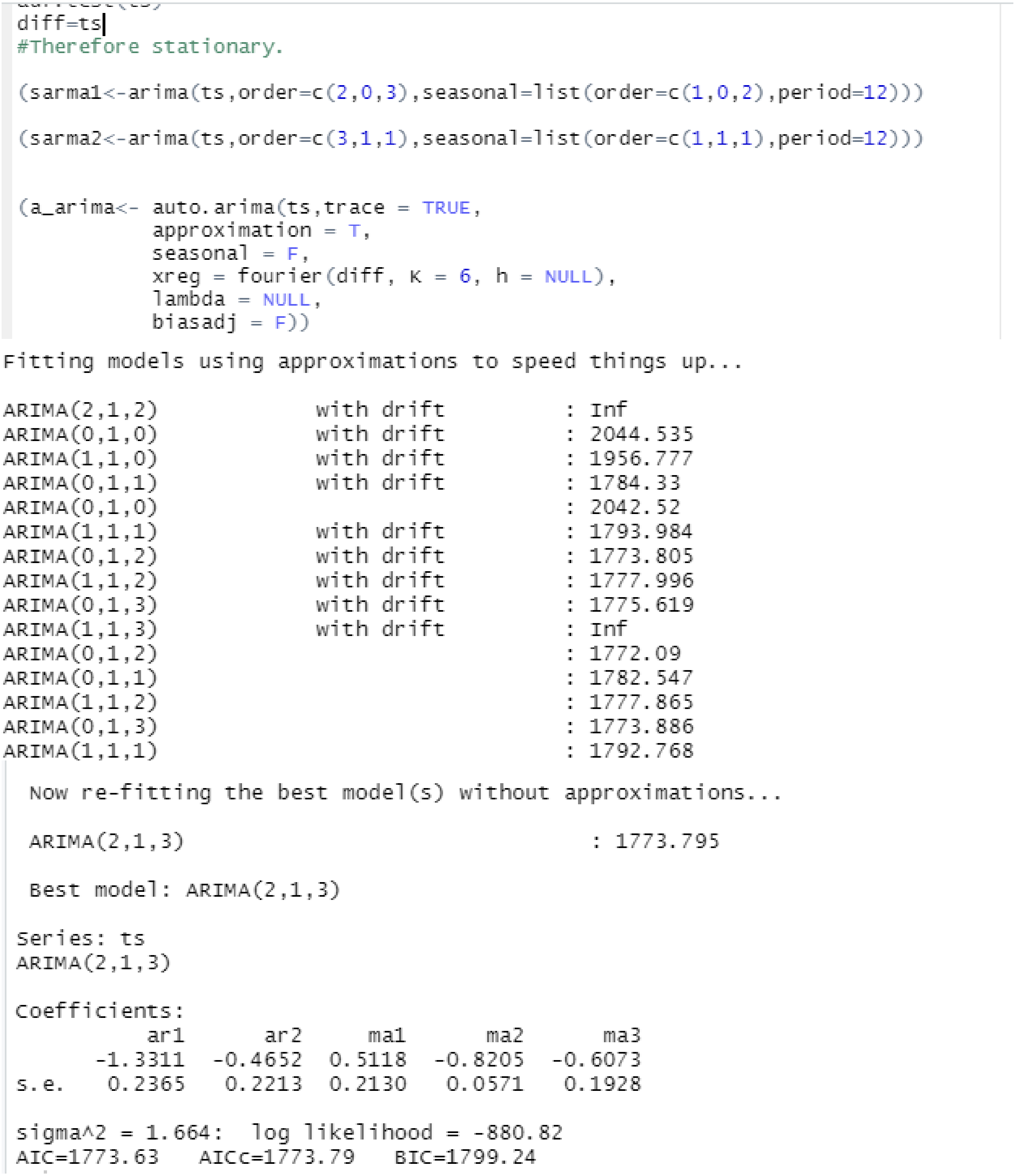
This model cannot be used , because it has higher value than the previous SARMA model . Also has the seasonality pattern is not certain we can use GARCH model and test to see if the AIC value is better along with ARMA model as well by ignoring the seasonal part.

GARCH model can be abbreviated as Generalized Auto-regressive conditional Heteroskedasticity models. However GARCH model is usually used to estimate value returns for stocks and so on , where trends is not known. We are using to test in our use-case to look at better AIC values. We are going to apply the seasonal ARMA-GARCH model using rugarch.

Steps to perform ARMA-GARCH:

- Check if the dataset is stationary or not like any other model.
- Identify the p and q order using ARIMA.
- To incorporate seasonality . Fourier terms are added .
- Check AIC values , Residuals and LB test and so on.

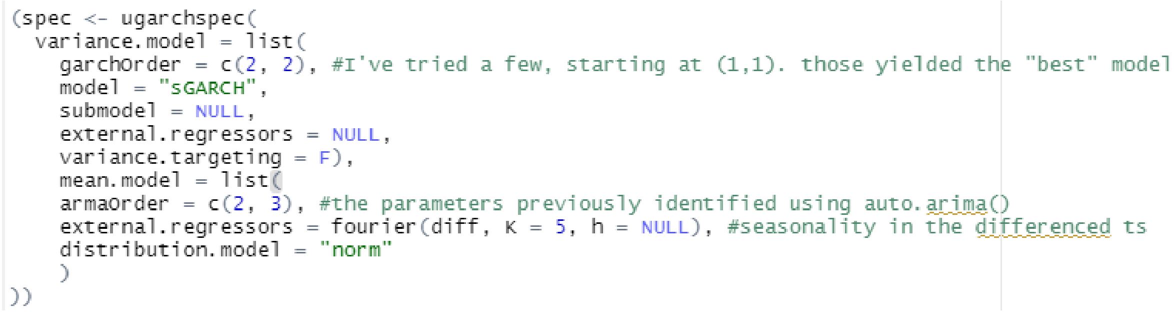
As we already know that the data is stationary , we can go about finding the p and q values from ACF and PACF plots or use auto.arima() in R.



```
diff=ts
#Therefore stationary.

(sarma1<-arima(ts,order=c(2,0,3),seasonal=list(order=c(1,0,2),period=12)))

(sarma2<-arima(ts,order=c(3,1,1),seasonal=list(order=c(1,1,1),period=12)))


(a_arima<- auto.arima(ts,trace = TRUE,
           approximation = T,
           seasonal = F,
           xreg = fourier(diff, K = 6, h = NULL),
           lambda = NULL,
           biasadj = F))
```

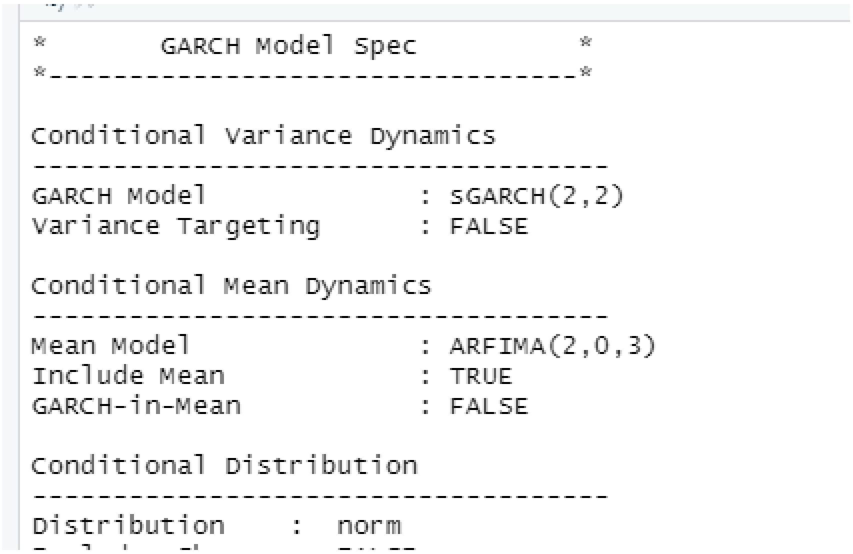
```
Fitting models using approximations to speed things up...

ARIMA(2,1,2)           with drift         : Inf
ARIMA(0,1,0)           with drift         : 2044.535
ARIMA(1,1,0)           with drift         : 1956.777
ARIMA(0,1,1)           with drift         : 1784.33
ARIMA(0,1,0)                              : 2042.52
ARIMA(1,1,1)           with drift         : 1793.984
ARIMA(0,1,2)           with drift         : 1773.805
ARIMA(1,1,2)           with drift         : 1777.996
ARIMA(0,1,3)           with drift         : 1775.619
ARIMA(1,1,3)           with drift         : Inf
ARIMA(0,1,2)                              : 1772.09
ARIMA(0,1,1)                              : 1782.547
ARIMA(1,1,2)                              : 1777.865
ARIMA(0,1,3)                              : 1773.886
ARIMA(1,1,1)                              : 1792.768

 Now re-fitting the best model(s) without approximations...

 ARIMA(2,1,3)                              : 1773.795

 Best model: ARIMA(2,1,3)

Series: ts
ARIMA(2,1,3)

Coefficients:
          ar1      ar2     ma1      ma2      ma3
      -1.3311  -0.4652  0.5118  -0.8205  -0.6073
s.e.   0.2365   0.2213  0.2130   0.0571   0.1928

sigma^2 = 1.664:  log likelihood = -880.82
AIC=1773.63   AICc=1773.79   BIC=1799.24
```

As we are not differencing the model we can consider ARMA(2,0,3) has the best model. Which is the and q value also found from the ACF and PACF plots.



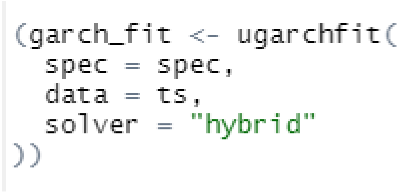
```
(spec <- ugarchspec(
  variance.model = list(
    garchOrder = c(2, 2), #I've tried a few, starting at (1,1). those yielded the "best" model
    model = "sGARCH",
    submodel = NULL,
    external.regressors = NULL,
    variance.targeting = F),
    mean.model = list(
    armaOrder = c(2, 3), #the parameters previously identified using auto.arima()
    external.regressors = fourier(diff, K = 5, h = NULL), #seasonality in the differenced ts
    distribution.model = "norm"
    )
))
```

Shows the specifications for GARCH model based on p and q values.



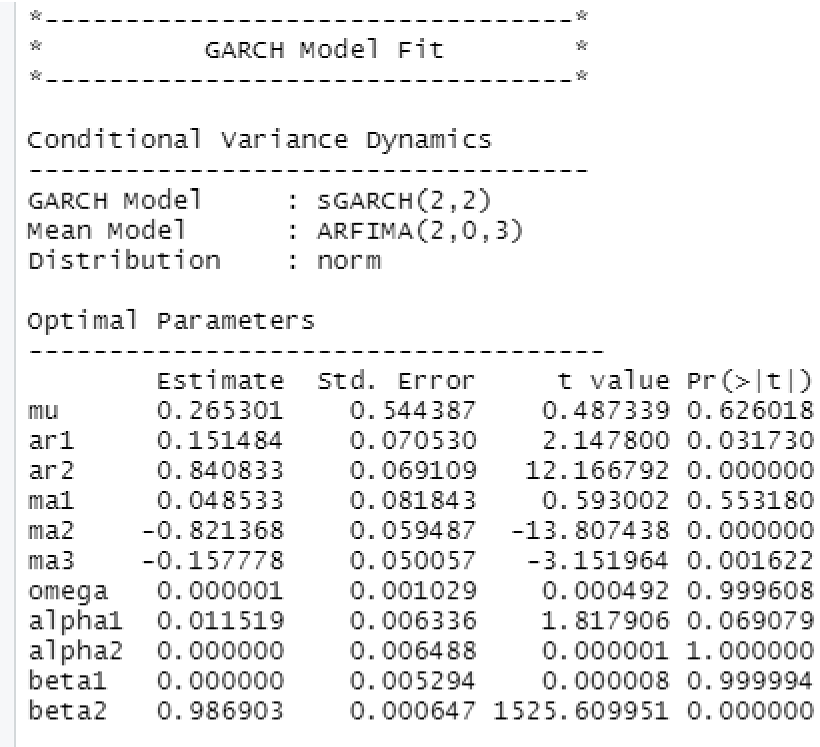
```
*       GARCH Model Spec          *
*---------------------------------*

Conditional Variance Dynamics
------------------------------------
GARCH Model              : sGARCH(2,2)
Variance Targeting       : FALSE

Conditional Mean Dynamics
------------------------------------
Mean Model               : ARFIMA(2,0,3)
Include Mean             : TRUE
GARCH-in-Mean            : FALSE

Conditional Distribution
------------------------------------
Distribution    :  norm
```

Shows the GARCH model specifications in the output.

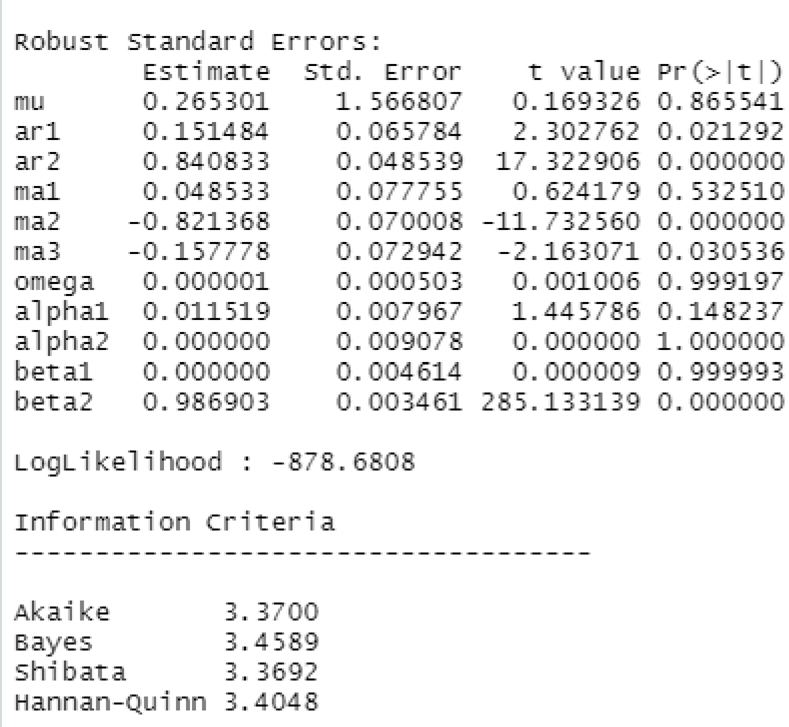


```
(garch_fit <- ugarchfit(
  spec = spec,
  data = ts,
  solver = "hybrid"
))
```

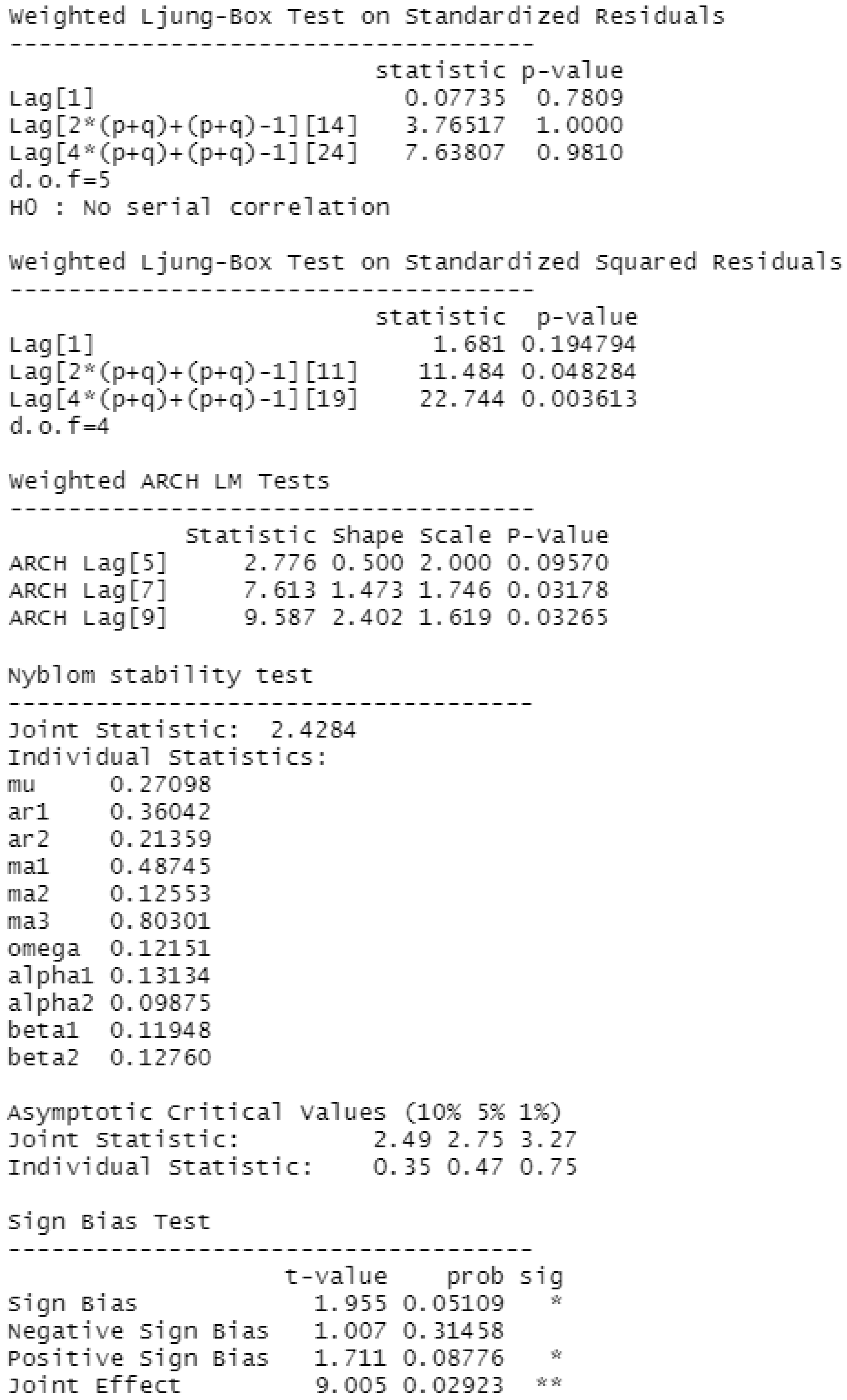
Fitting the GARCH model with the dataset:



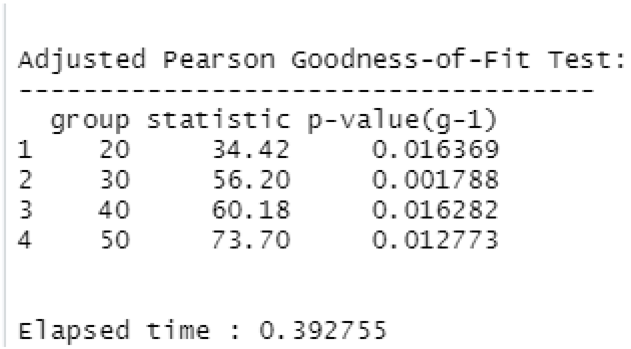
```
*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(2,2)
Mean Model      : ARFIMA(2,0,3)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error     t value Pr(>|t|)
mu      0.265301    0.544387    0.487339 0.626018
ar1     0.151484    0.070530    2.147800 0.031730
ar2     0.840833    0.069109   12.166792 0.000000
ma1     0.048533    0.081843    0.593002 0.553180
ma2    -0.821368    0.059487  -13.807438 0.000000
ma3    -0.157778    0.050057   -3.151964 0.001622
omega   0.000001    0.001029    0.000492 0.999608
alpha1  0.011519    0.006336    1.817906 0.069079
alpha2  0.000000    0.006488    0.000001 1.000000
beta1   0.000000    0.005294    0.000008 0.999994
beta2   0.986903    0.000647 1525.609951 0.000000
```



```
Robust Standard Errors:
        Estimate  Std. Error    t value Pr(>|t|)
mu      0.265301    1.566807   0.169326 0.865541
ar1     0.151484    0.065784   2.302762 0.021292
ar2     0.840833    0.048539  17.322906 0.000000
ma1     0.048533    0.077755   0.624179 0.532510
ma2    -0.821368    0.070008 -11.732560 0.000000
ma3    -0.157778    0.072942  -2.163071 0.030536
omega   0.000001    0.000503   0.001006 0.999197
alpha1  0.011519    0.007967   1.445786 0.148237
alpha2  0.000000    0.009078   0.000000 1.000000
beta1   0.000000    0.004614   0.000009 0.999993
beta2   0.986903    0.003461 285.133139 0.000000

LogLikelihood : -878.6808

Information Criteria
------------------------------------

Akaike       3.3700
Bayes        3.4589
Shibata      3.3692
Hannan-Quinn 3.4048
```



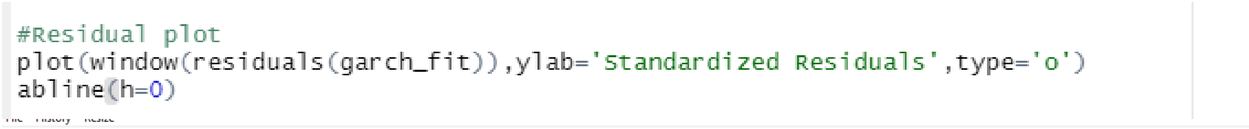
```
Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic p-value
Lag[1]                     0.07735  0.7809
Lag[2*(p+q)+(p+q)-1][14]   3.76517  1.0000
Lag[4*(p+q)+(p+q)-1][24]   7.63807  0.9810
d.o.f=5
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                         statistic  p-value
Lag[1]                       1.681 0.194794
Lag[2*(p+q)+(p+q)-1][11]    11.484 0.048284
Lag[4*(p+q)+(p+q)-1][19]    22.744 0.003613
d.o.f=4

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[5]     2.776 0.500 2.000 0.09570
ARCH Lag[7]     7.613 1.473 1.746 0.03178
ARCH Lag[9]     9.587 2.402 1.619 0.03265

Nyblom stability test
------------------------------------
Joint Statistic:  2.4284
Individual Statistics:
mu     0.27098
ar1    0.36042
ar2    0.21359
ma1    0.48745
ma2    0.12553
ma3    0.80301
omega  0.12151
alpha1 0.13134
alpha2 0.09875
beta1  0.11948
beta2  0.12760

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         2.49 2.75 3.27
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
                   t-value    prob sig
Sign Bias            1.955 0.05109   *
Negative Sign Bias   1.007 0.31458
Positive Sign Bias   1.711 0.08776   *
Joint Effect         9.005 0.02923  **
```



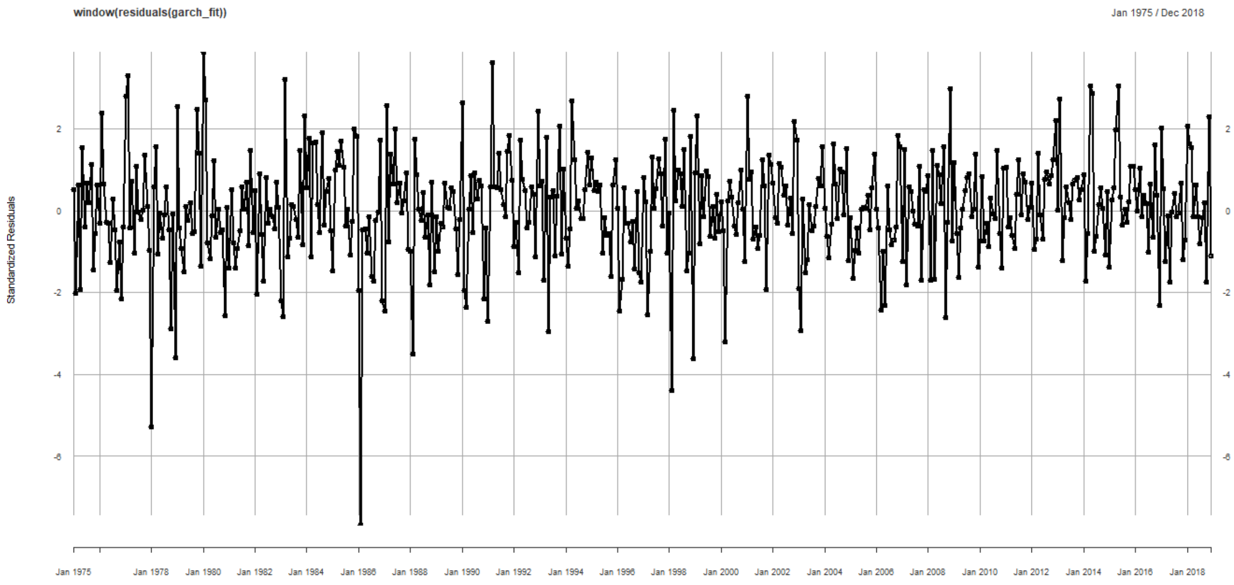
```
Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     34.42     0.016369
2    30     56.20     0.001788
3    40     60.18     0.016282
4    50     73.70     0.012773


Elapsed time : 0.392755
```

Shows the output of the GARCH model when ran of the dataset. Some of the observations we can see that and compare with the SARMA model . We can view various optimal parameters and their estimate and standard error as well. The LB test has values for p nothing less than 0.05 so we can say that null hypothesis is rejected and may assume correlation being present in the dataset. However LB test on Standardized squared residuals yield one of the p value closer to 0. The Arch LM tests has p values for lags 7 and 9 closer to zero. We can observe that the log-likelihood for SARMA is smaller than GARCH . Although, it is well noted that the higher the log-likelihood, the better the model. The GARCH model has higher log-likelihood when compared sarima but no much difference between their values. While the same is opposite for AIC value , where smaller the AIC value is the best model.

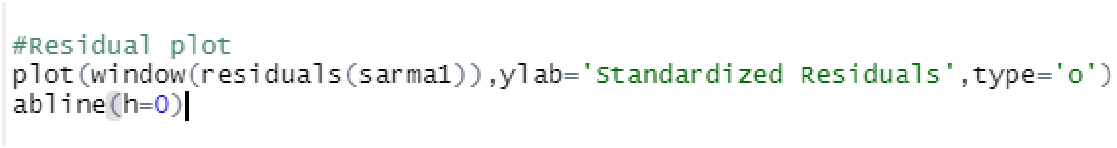
4.5 Residual Plots:



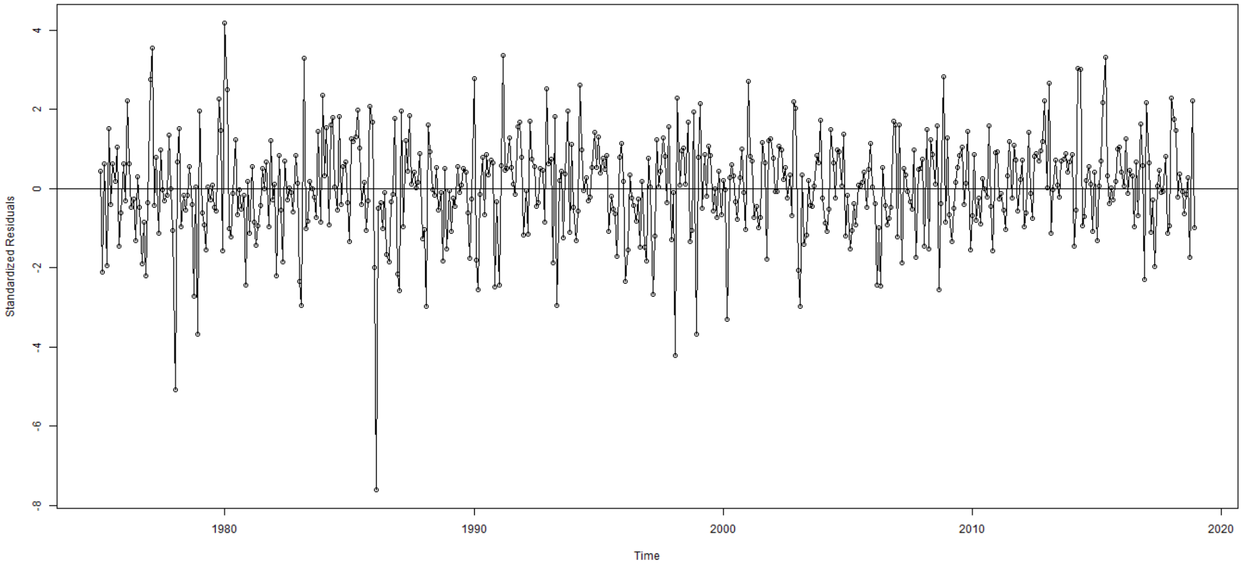
```
#Residual plot
plot(window(residuals(garch_fit)),ylab='Standardized Residuals',type='o')
abline(h=0)
```



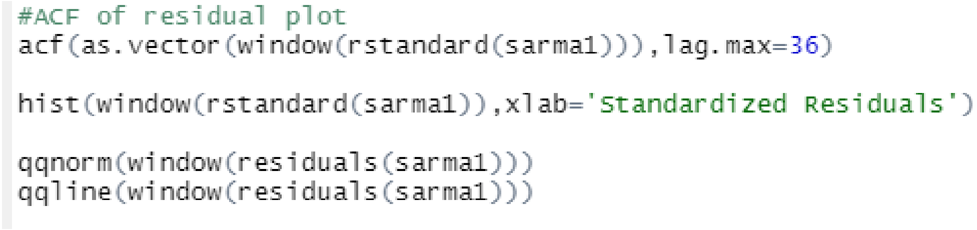
Shows the residual plot for GARCH.



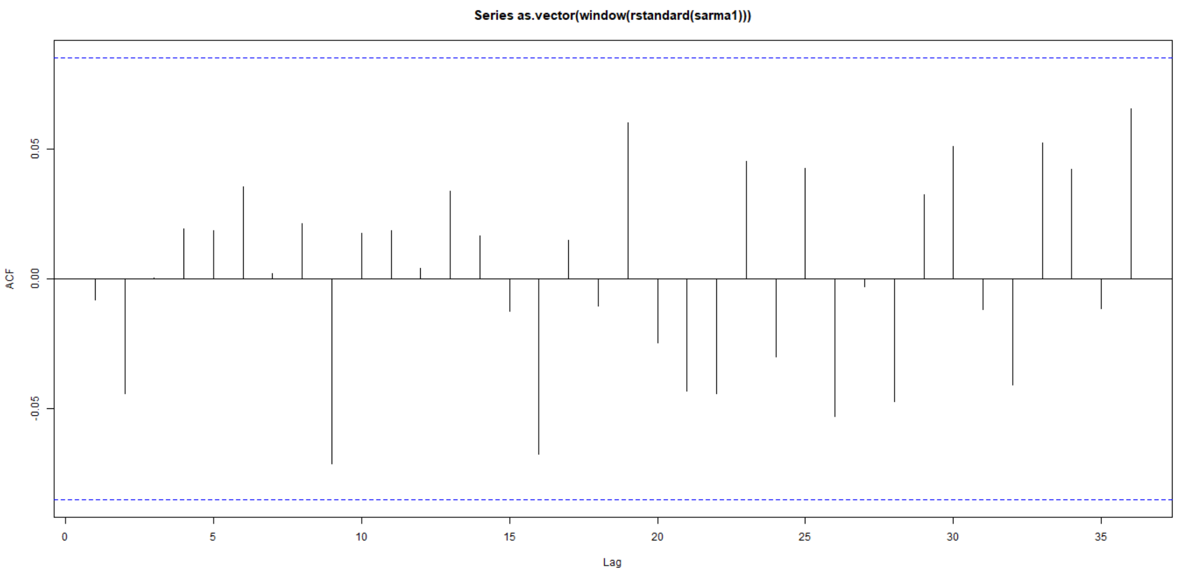
```
#Residual plot
plot(window(residuals(sarma1)),ylab='Standardized Residuals',type='o')
abline(h=0)
```

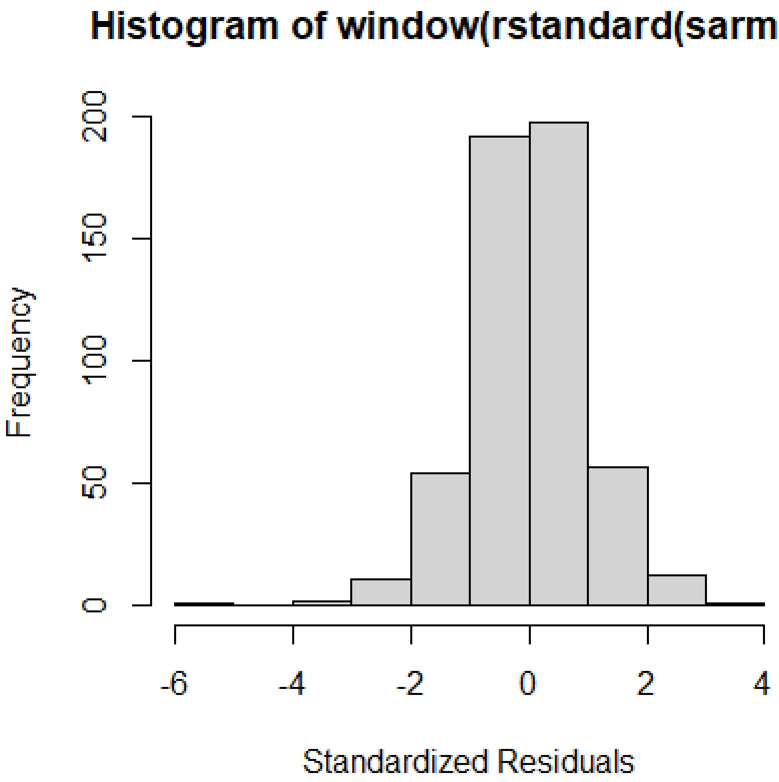


Shows the Residual plot for SARMA.There is no significant change in the residual analysis after adopting the GARCH model. The pattern looks almost constant. Further Analysis can be done with SARMA model:

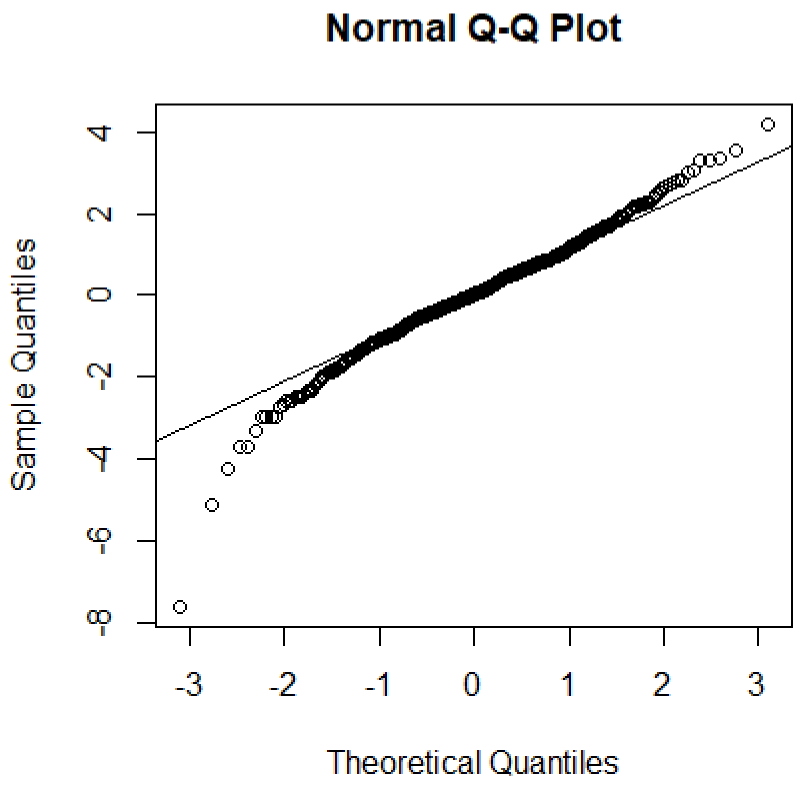


```
#ACF of residual plot
acf(as.vector(window(rstandard(sarma1))),lag.max=36)

hist(window(rstandard(sarma1)),xlab='Standardized Residuals')

qqnorm(window(residuals(sarma1)))
qqline(window(residuals(sarma1)))
```

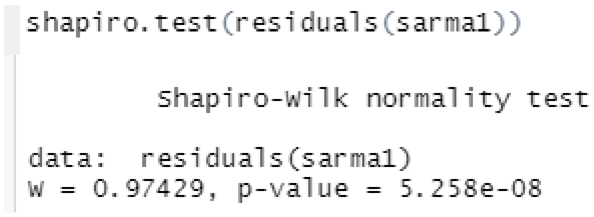




The plot demonstrates the histogram of the residuals. the shape of the curve is almost bell shaped and form the normal distribution.

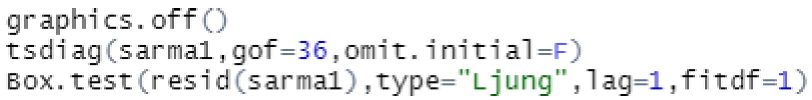


From the qq plot we can see slightly towards the ends there is outliers and deviation from the reference line for theoretical and sample quantiles. Though there is a presence of outliers it is not as far from the reference line.

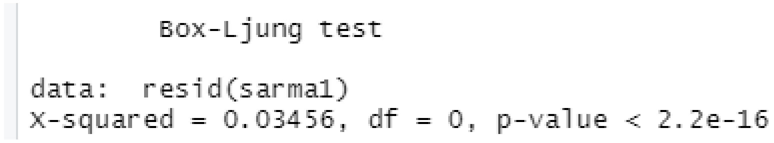


```
shapiro.test(residuals(sarma1))

        Shapiro-Wilk normality test

data:  residuals(sarma1)
W = 0.97429, p-value = 5.258e-08
```

Shapiro-Wilk test yields a W value of 0.9778 and p value of 0.00000000525 which is much smaller than 0.05 and very close to zero. Thus data is non-normal.

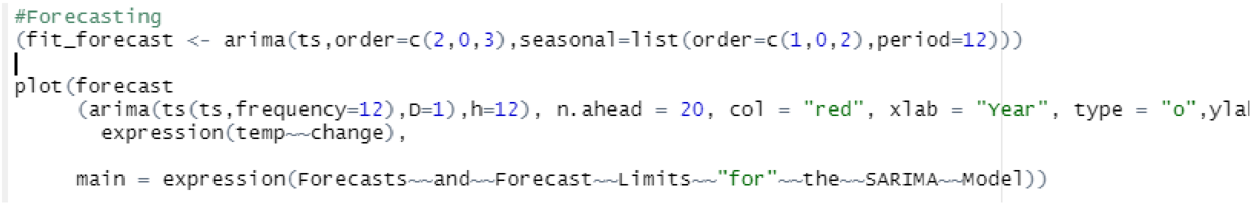


```
graphics.off()
tsdiag(sarma1,gof=36,omit.initial=F)
Box.test(resid(sarma1),type="Ljung",lag=1,fitdf=1)
```

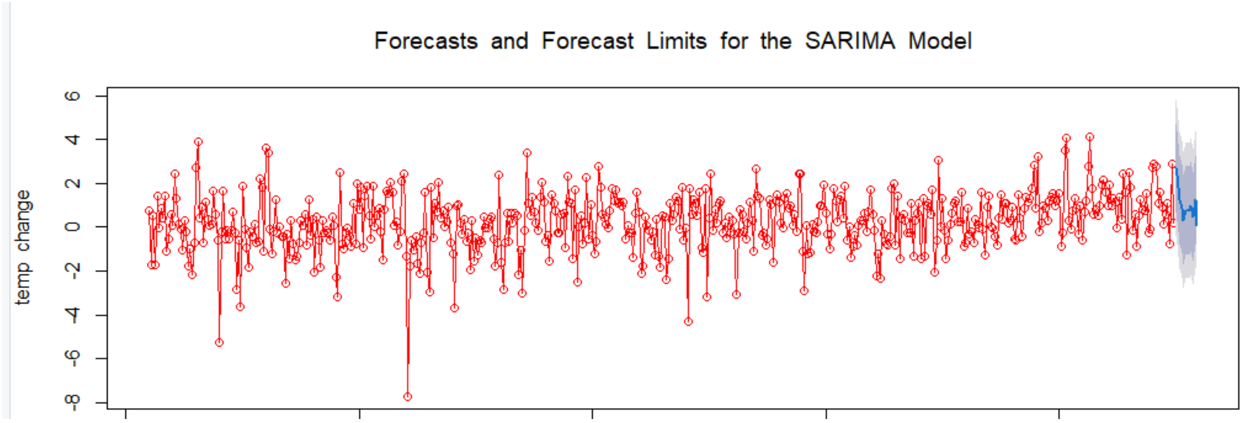


```
        Box-Ljung test

data:  resid(sarma1)
X-squared = 0.03456, df = 0, p-value < 2.2e-16
```

LB test will test if there is autocorrelation in the time series data. P value is very small and close to zero therefore, it has grabbed the dependence in the time series.



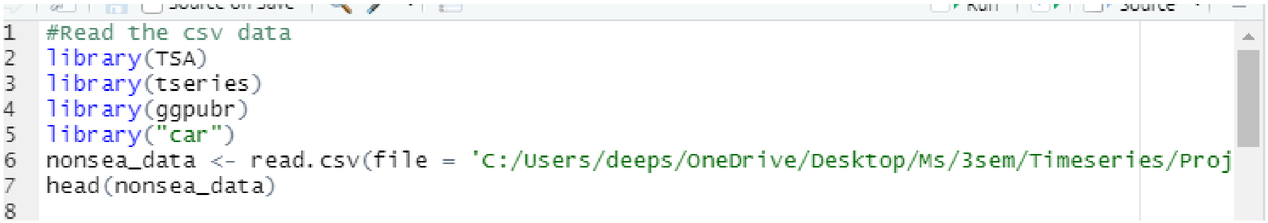
```
#Forecasting
(fit_forecast <- arima(ts,order=c(2,0,3),seasonal=list(order=c(1,0,2),period=12)))

plot(forecast
      (arima(ts(ts,frequency=12),D=1),h=12), n.ahead = 20, col = "red", xlab = "Year", type = "o",ylal
        expression(temp~~change),

      main = expression(Forecasts~~and~~Forecast~~Limits~~"for"~~the~~SARIMA~~Model))
```



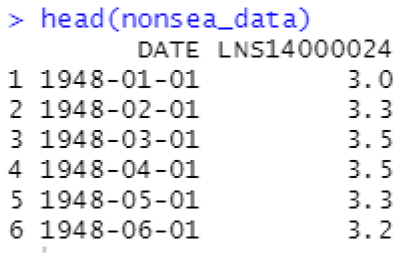
Shows the forecasting for the time series with the original dataset. It gives a good forecasting and accuracy.

5. Arima Non-Seasonal Time series modeling

5.1 Data Reading and Preparation:



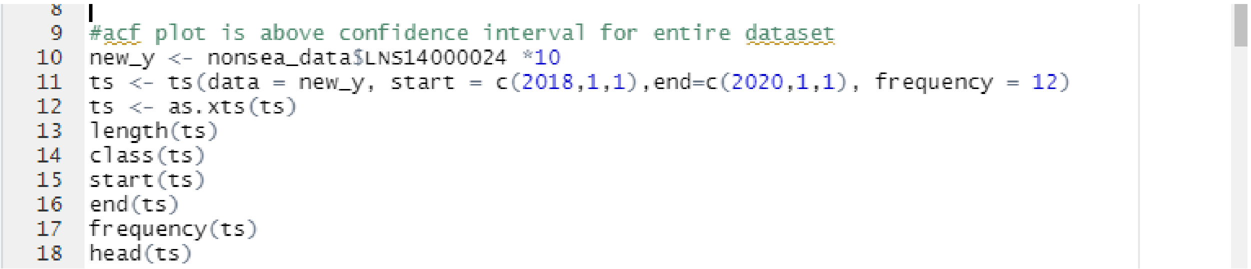
```
#Read the csv data
library(TSA)
library(tseries)
library(ggpubr)
library("car")
nonsea_data <- read.csv(file = 'C:/Users/deeps/OneDrive/Desktop/Ms/3sem/Timeseries/Proj
head(nonsea_data)
```



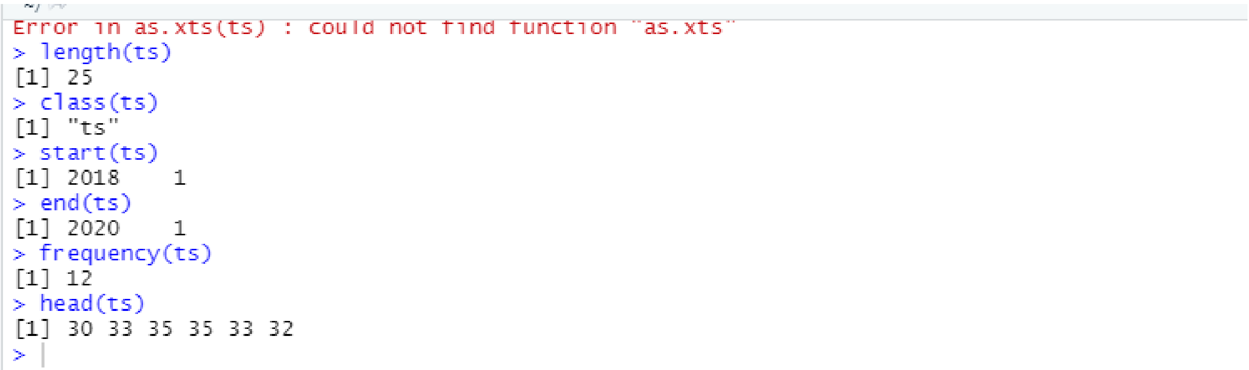
```
> head(nonsea_data)
        DATE LNS14000024
1 1948-01-01         3.0
2 1948-02-01         3.3
3 1948-03-01         3.5
4 1948-04-01         3.5
5 1948-05-01         3.3
6 1948-06-01         3.2
```

End to End Time series project for seasonal and non-seasonal datasets

Shows Reading the csv data in R . We can see that dataset started with Jan 1948 and has the value of unemployment rate stored in LNS140000024 variable.



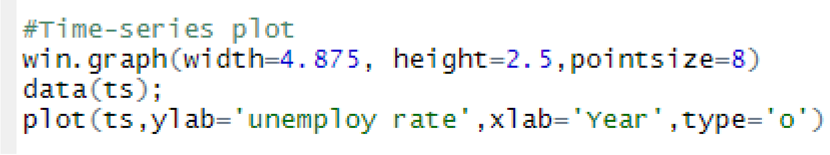
```
#acf plot is above confidence interval for entire dataset
new_y <- nonsea_data$LNS14000024 *10
ts <- ts(data = new_y, start = c(2018,1,1),end=c(2020,1,1), frequency = 12)
ts <- as.xts(ts)
length(ts)
class(ts)
start(ts)
end(ts)
frequency(ts)
head(ts)
```



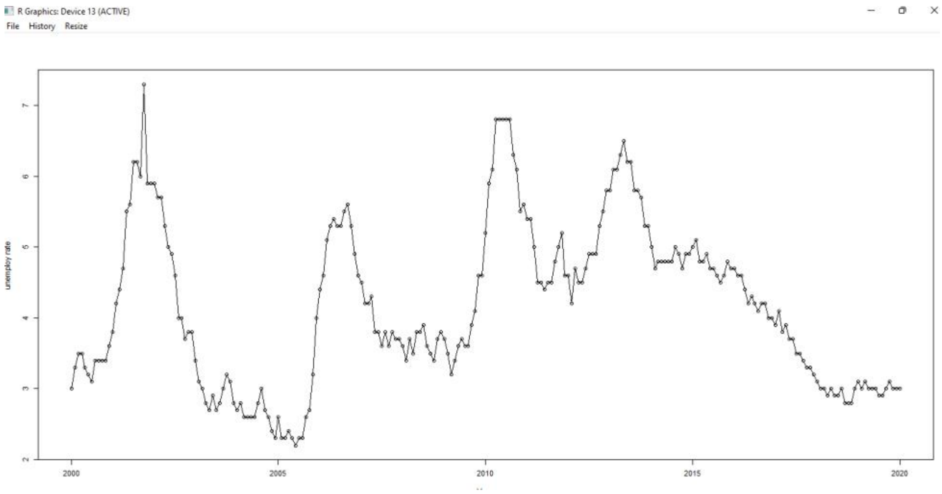
```
Error in as.xts(ts) : could not find function "as.xts"
> length(ts)
[1] 25
> class(ts)
[1] "ts"
> start(ts)
[1] 2018    1
> end(ts)
[1] 2020    1
> frequency(ts)
[1] 12
> head(ts)
[1] 30 33 35 35 33 32
>
```

Shows that the dataset for time-series is sampled between 2018 and 2020 to find the recent data and plot them and analyze the time series. It has about 25 data points and starting with Jan 2018 and ending with Jan 2020 .

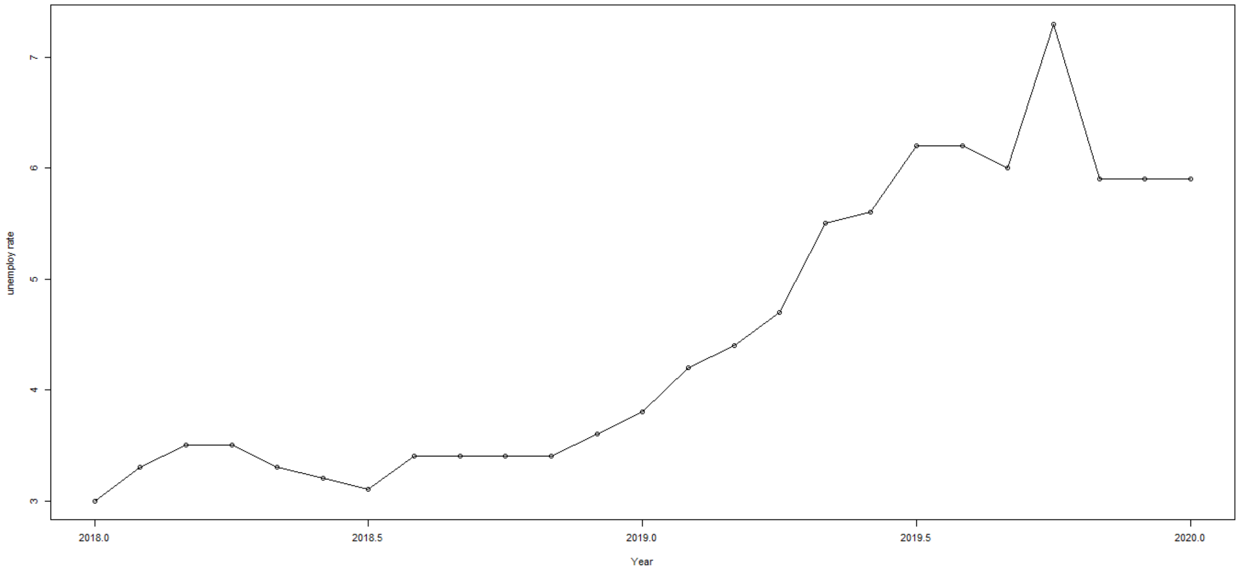
5.2 Time-series analysis:



```
#Time-series plot
win.graph(width=4.875, height=2.5,pointsize=8)
data(ts);
plot(ts,ylab='unemploy rate',xlab='Year',type='o')
```



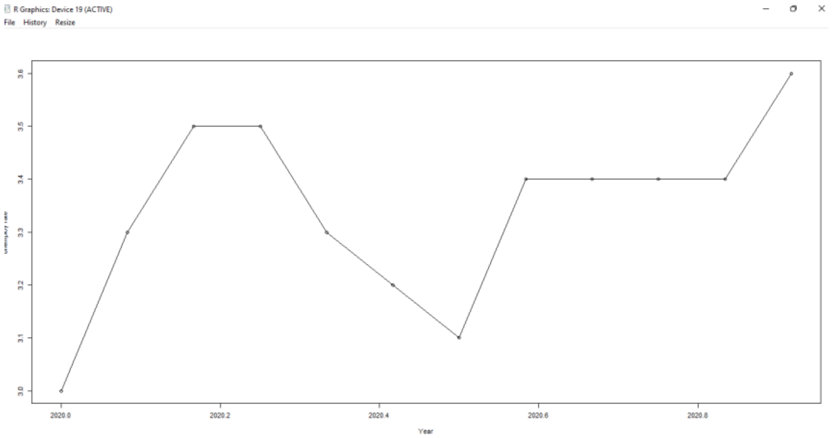
We can see that the time series plot between 2000 and 2020 is random in nature and is not stationary just by looking at it.

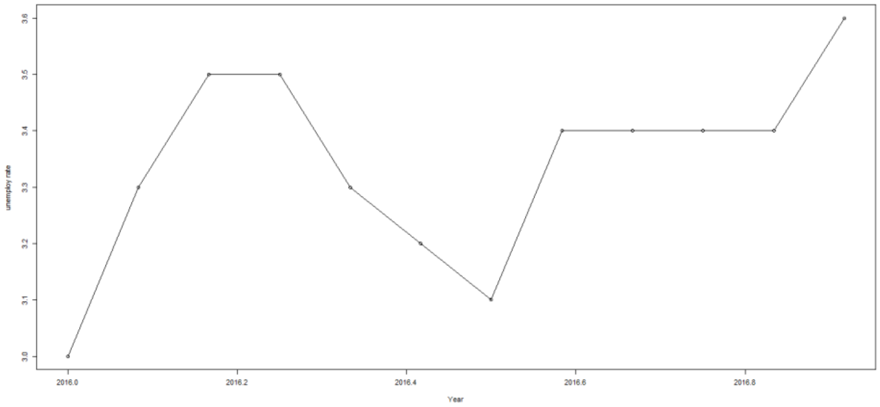


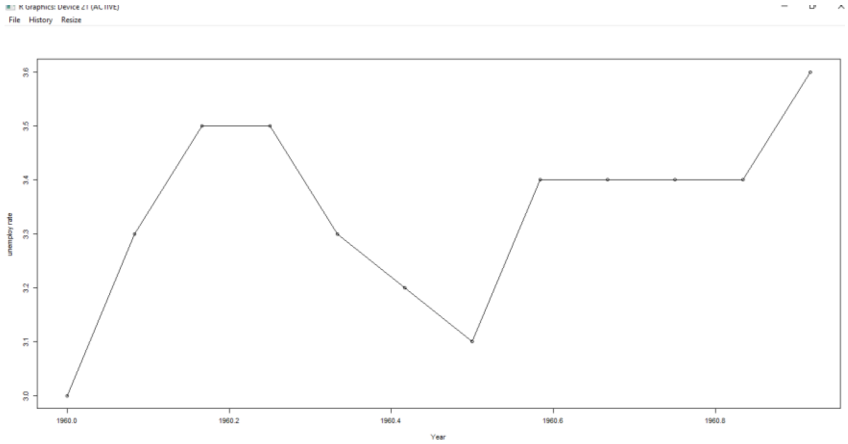
This is the time-series plot between 2018 and 2020 sampling and can see a clear upward trend . This is non-stationary in nature and will require further differentiation to make it stationary.

5.3 Comparing various yearly trends of different year values:

End to End Time series project for seasonal and non-seasonal datasets

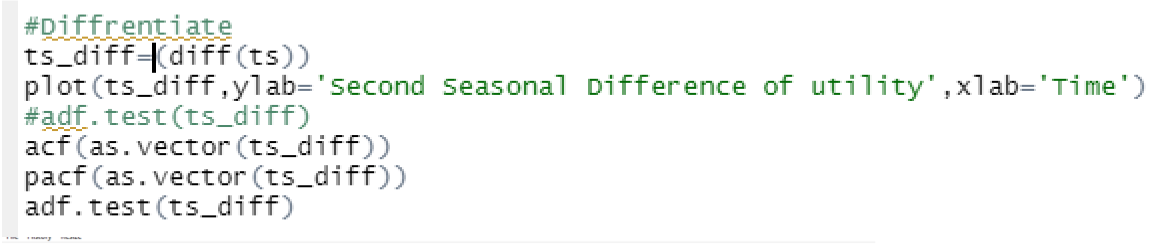




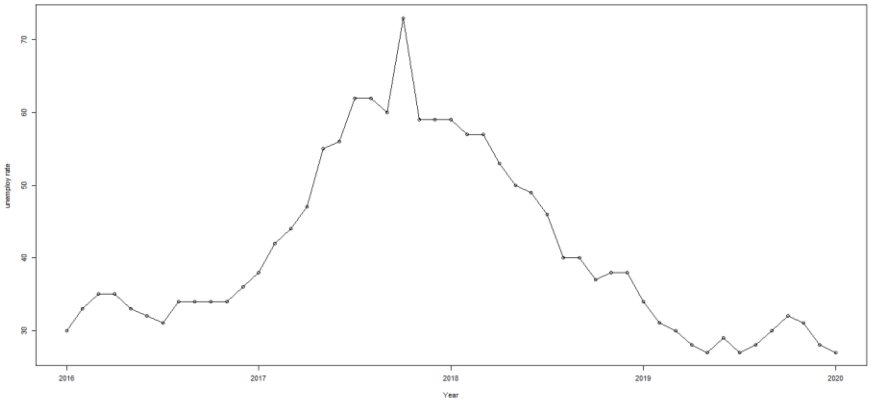


From the above time series plot we can conclude that , the trend within the year values for 1960,2016 and 2020 are similar . We can observe that during start of the year in January the unemployment rate increases and becomes constant during February , March and then decreases sharply post April. Then in mid of the year it increases to a certain level and attains constant until late/end of the year. Clearly we can see some pattern when we do time series plot within a single year. It can be concluded that unemployment rate is higher during winter months and decreased post April which is summer season. However the yearly data may be seasonal , but if we try to compare different years there is no seasonality found and that is considered for further analysis throughout the report(Which is between 2018 and 2020).

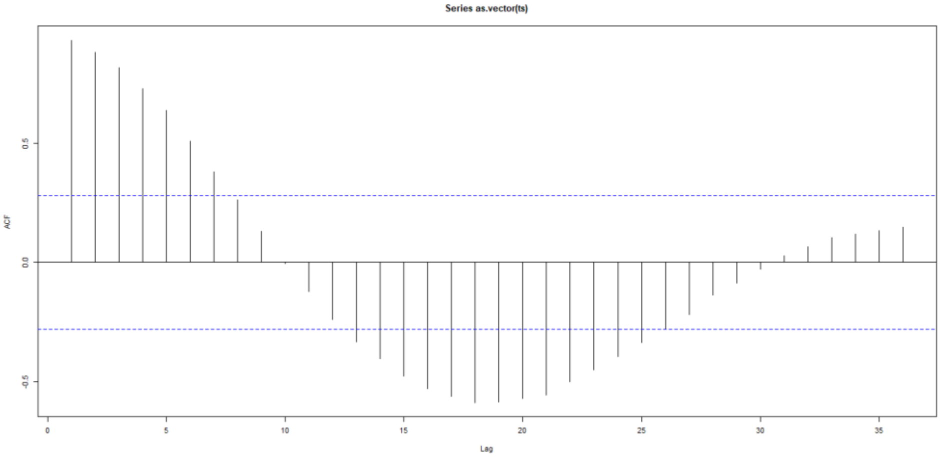
5.4 ACF Plot analysis for sample between 2016 and 2020:



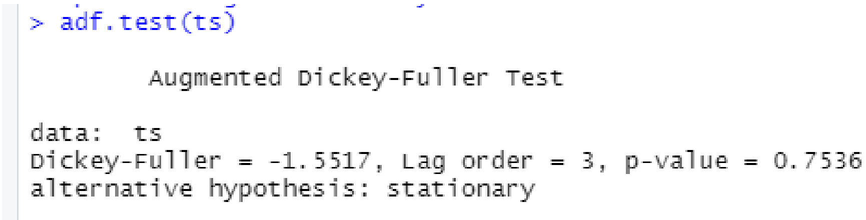
```
#Diffrentiate
ts_diff=(diff(ts))
plot(ts_diff,ylab='Second Seasonal Difference of utility',xlab='Time')
#adf.test(ts_diff)
acf(as.vector(ts_diff))
pacf(as.vector(ts_diff))
adf.test(ts_diff)
```



Time series plot is not stationary.



Shows the initial ACF plot and we can see that before lag 25 all are significant and being having a no trend it needs to be differentiated before performing any analysis .

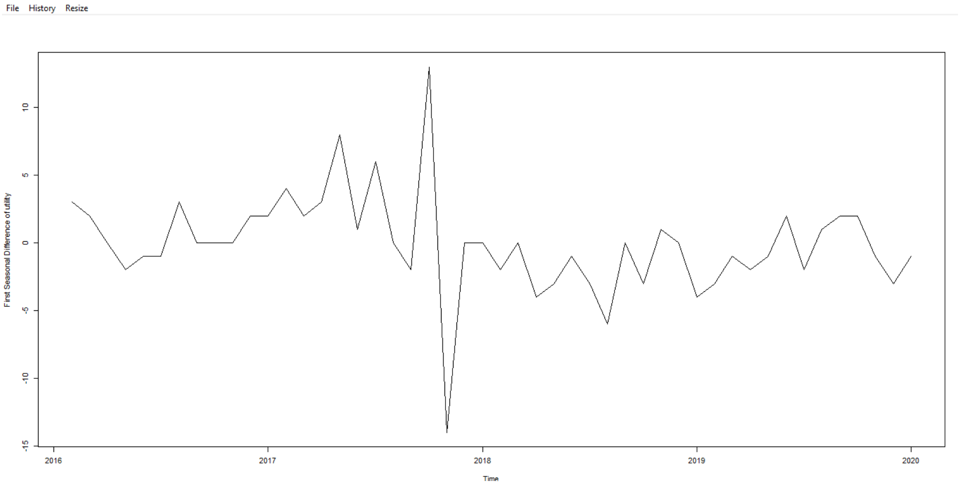


```
> adf.test(ts)

        Augmented Dickey-Fuller Test

data:  ts
Dickey-Fuller = -1.5517, Lag order = 3, p-value = 0.7536
alternative hypothesis: stationary
```

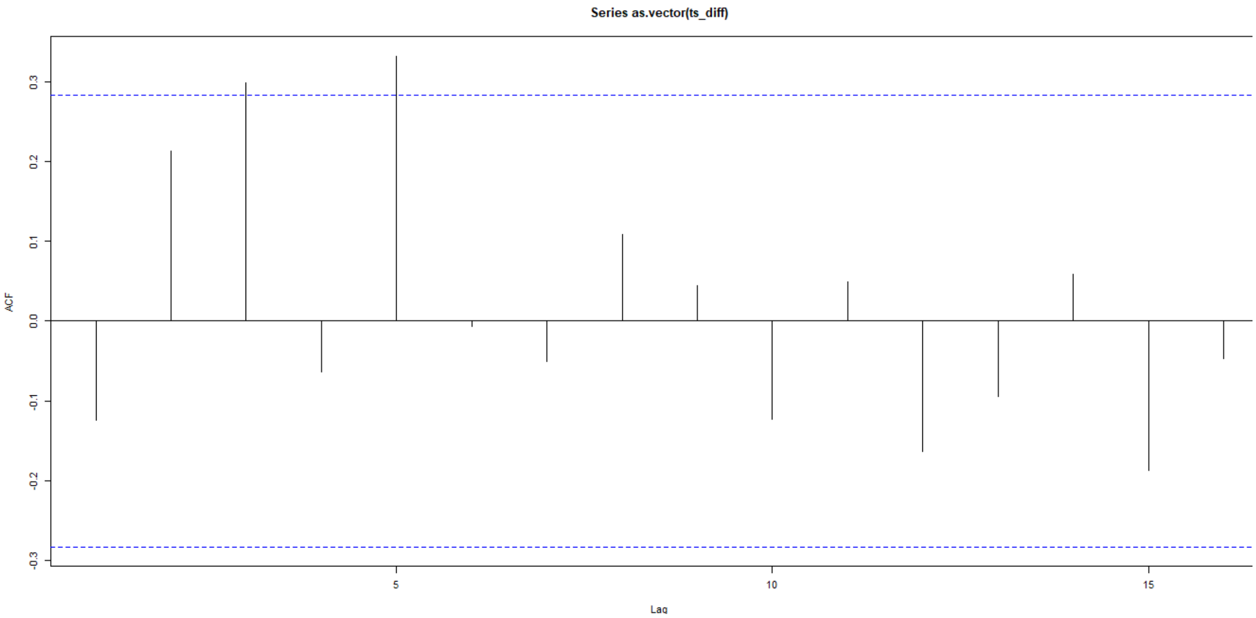
On performing the dickey-fuller test , we can see that p values > 0.05 and is clearly non-stationary in nature.

5.5 Converting non-stationary to stationary model:

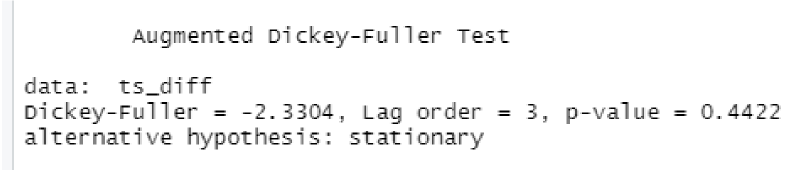
First lets differentiate the time series dataset , view the time series plot,acf and perform the test for stationary . Even then if the dickey filler test gives p values higher we need to further differentiate the dataset or apply log transformation and then perform further analysis.



The time series plot on differentiation has almost become stationary other than the middle most part , almost near 2018 . We can confirm the same with dickey filler test and find the p value to check its stationary.



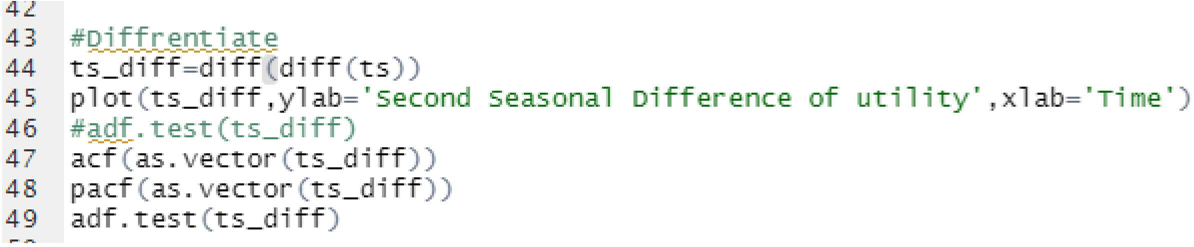
We can see that the ACF plot , has changed significantly . there are only 2 significant values within 5 lags and rest are within the confidence interval.



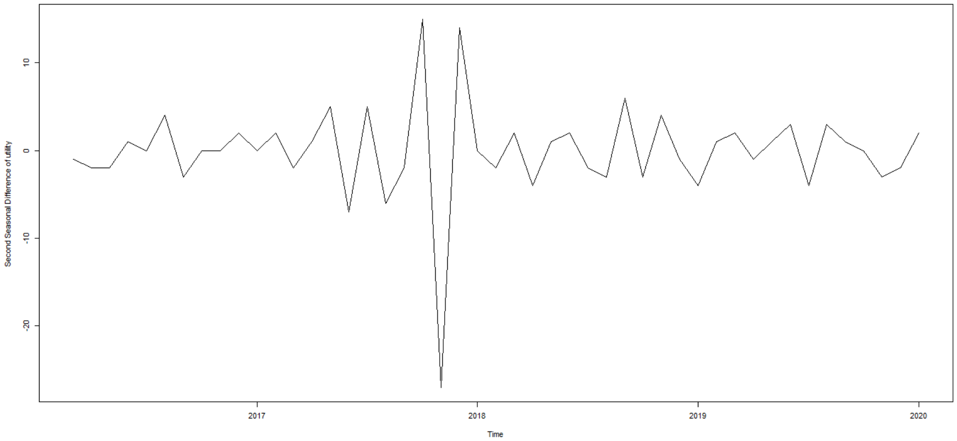
```
	Augmented Dickey-Fuller Test

data:  ts_diff
Dickey-Fuller = -2.3304, Lag order = 3, p-value = 0.4422
alternative hypothesis: stationary
```

The test has shown has p value reduced from before time series without differencing. However it is not small enough to make the series stationary. This clearly shows that time series is still non-stationary.

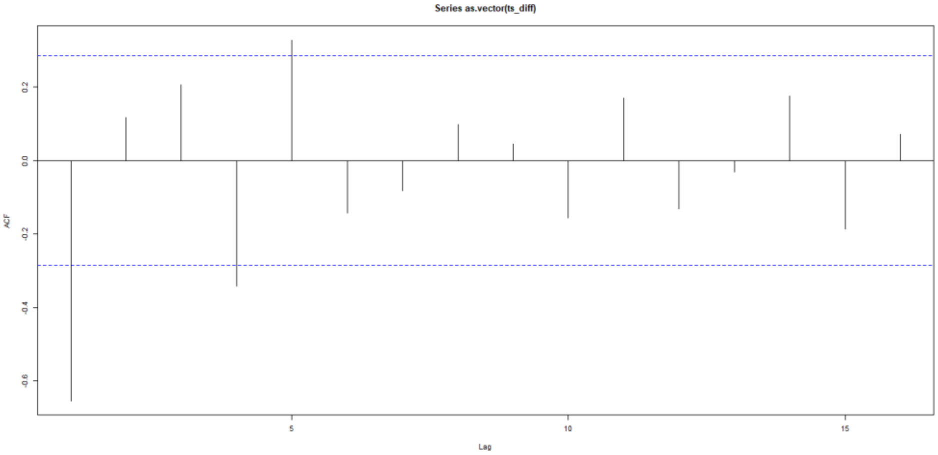
On applying double differentiation.



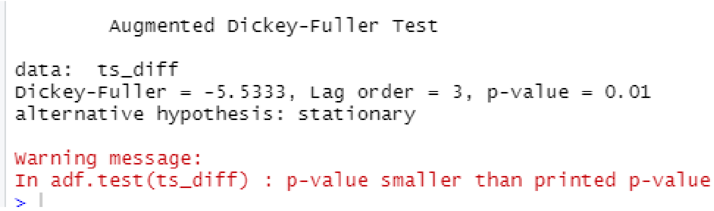
```
#Diffrentiate
ts_diff=diff(diff(ts))
plot(ts_diff,ylab='Second Seasonal Difference of utility',xlab='Time')
#adf.test(ts_diff)
acf(as.vector(ts_diff))
pacf(as.vector(ts_diff))
adf.test(ts_diff)
```



The time series plot has changes near the year 2018 almost to a stationary looking trend , previously it had dropped down significantly . Only the DF test can confirm this.



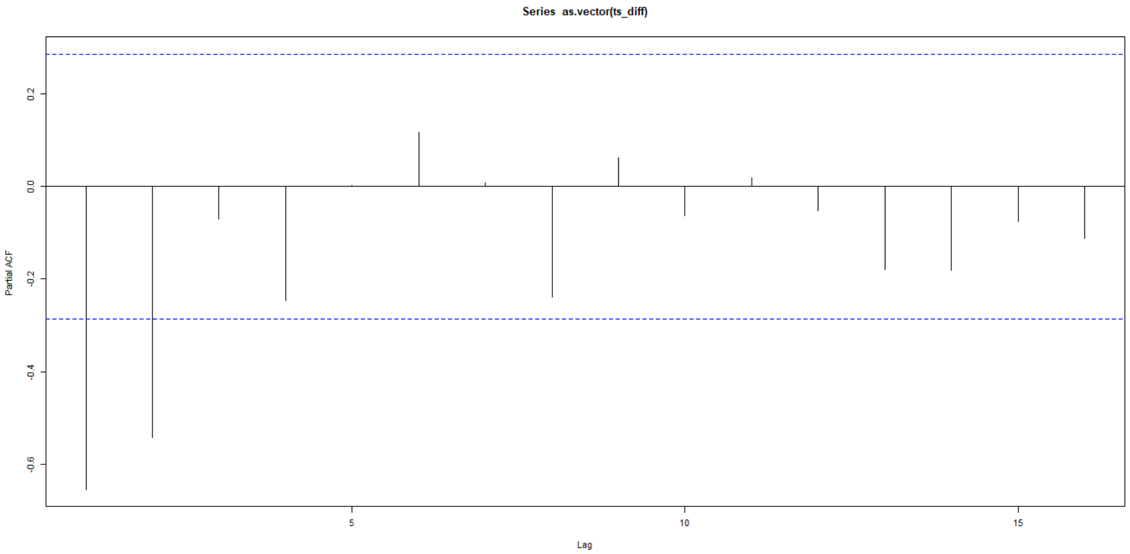
The ACF plot shows 3 significant before lags 5 and this data will be used for further designing t he arima models. There is no seasonality found in the ACF plots and thus it can be considered as a non-seasonal data for this sample. 1,4,5 have significant for the lag values .We can use MA(1/4/5) models to model the time series.



```
        Augmented Dickey-Fuller Test

data:  ts_diff
Dickey-Fuller = -5.5333, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(ts_diff) : p-value smaller than printed p-value
> 
```

The DF test confirms that it is stationary as p value < 0.05 and thus can be used for further analysis.

5.6 PACF Plot analysis for stationary dataset:



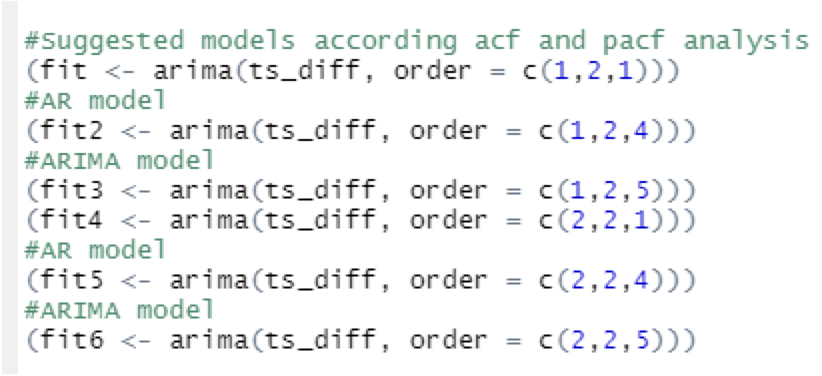
From the PACF plot we can see that 1,2 have strong correlation for the lags. So it can be AR(1/2).

5.7 Modeling and Parameter estimation:

From the above , ACF and PACF plot we can observe that we will need ARIMA model and not ARMA because the time series is double differentiated .Also, it has both AR and MA component based on strong auto-correlation for the lags thus suggested models are,

ARIMA(1,2,1)
ARIMA(1,2,4)
ARIMA(1,2,5)
ARIMA(2,2,1)
ARIMA(2,2,4)
ARIMA(2,2,5)

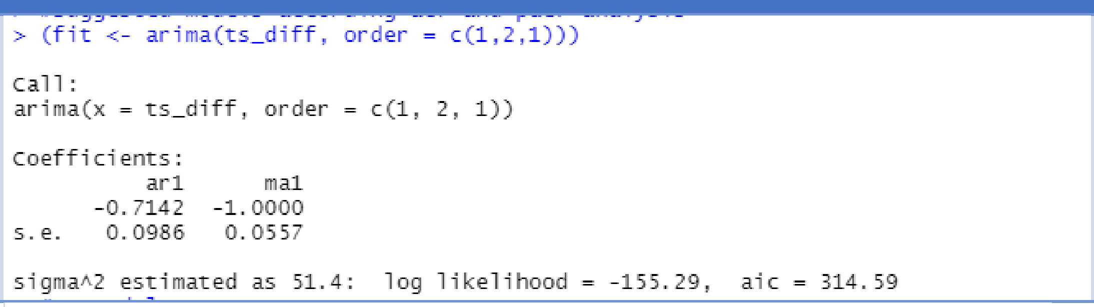
Where the ARIMA (PACF,Num_Diffrentation,ACF) model have the below format for the parameters.Coefficients for various models:

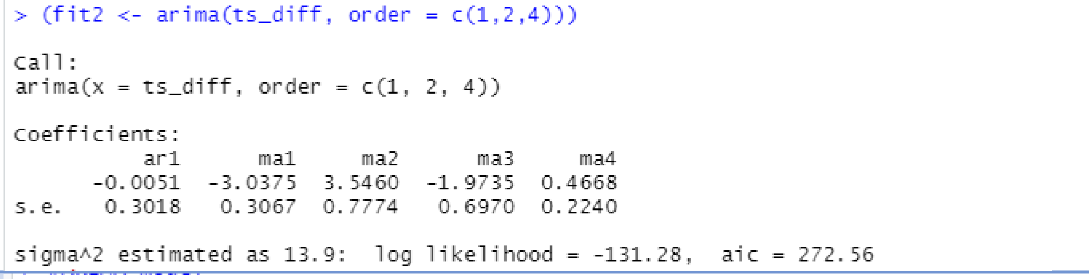


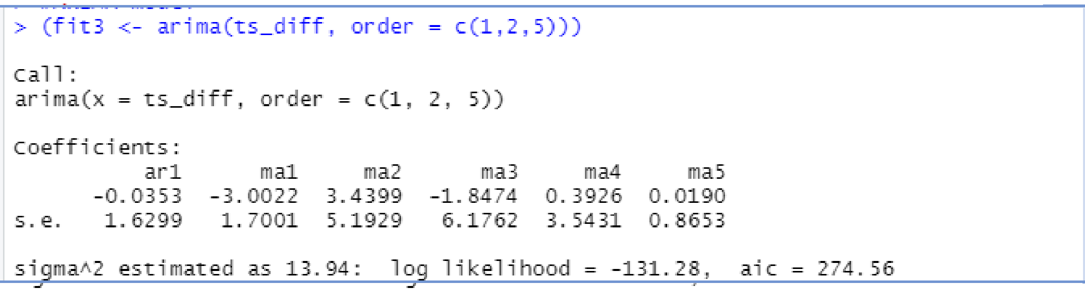
```
#Suggested models according acf and pacf analysis
(fit <- arima(ts_diff, order = c(1,2,1)))
#AR model
(fit2 <- arima(ts_diff, order = c(1,2,4)))
#ARIMA model
(fit3 <- arima(ts_diff, order = c(1,2,5)))
(fit4 <- arima(ts_diff, order = c(2,2,1)))
#AR model
(fit5 <- arima(ts_diff, order = c(2,2,4)))
#ARIMA model
(fit6 <- arima(ts_diff, order = c(2,2,5)))
```

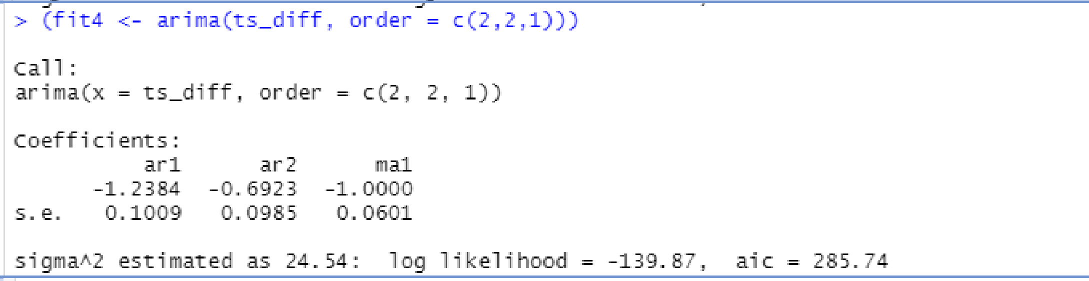
End to End Time series project for seasonal and non-seasonal datasets

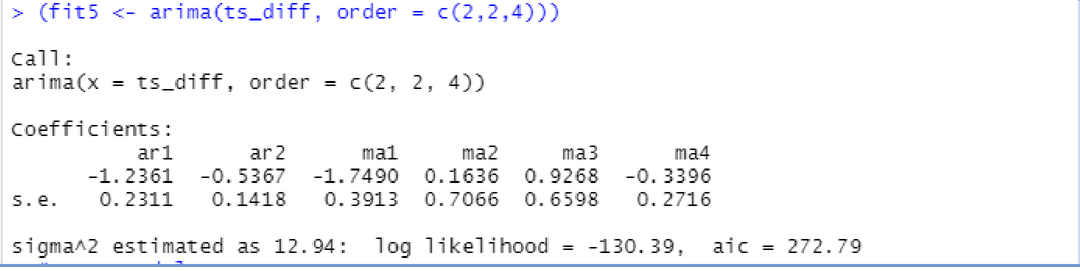
Model	Coefficients and AIC values

ARIMA(1,2,1)	
	`> (fit <- arima(ts_diff, order = c(1,2,1)))`
	
	`Call:`
	`arima(x = ts_diff, order = c(1, 2, 1))`
	
	`Coefficients:`
	`          ar1      ma1`
	`      -0.7142  -1.0000`
	`s.e.   0.0986   0.0557`
	
	`sigma^2 estimated as 51.4:  log likelihood = -155.29,  aic = 314.59`

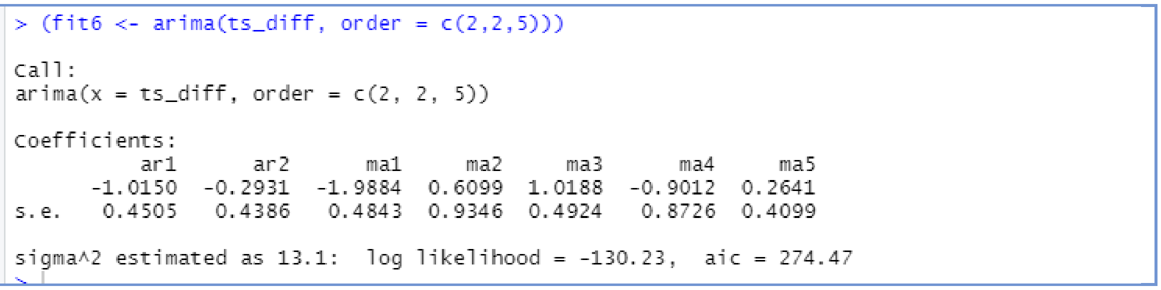
ARIMA(1,2,4)	
	`> (fit2 <- arima(ts_diff, order = c(1,2,4)))`
	
	`Call:`
	`arima(x = ts_diff, order = c(1, 2, 4))`
	
	`Coefficients:`
	`          ar1      ma1     ma2      ma3     ma4`
	`      -0.0051  -3.0375  3.5460  -1.9735  0.4668`
	`s.e.   0.3018   0.3067  0.7774   0.6970  0.2240`
	
	`sigma^2 estimated as 13.9:  log likelihood = -131.28,  aic = 272.56`

ARIMA(1,2,5)	
	`> (fit3 <- arima(ts_diff, order = c(1,2,5)))`
	
	`Call:`
	`arima(x = ts_diff, order = c(1, 2, 5))`
	
	`Coefficients:`
	`          ar1      ma1     ma2      ma3     ma4     ma5`
	`      -0.0353  -3.0022  3.4399  -1.8474  0.3926  0.0190`
	`s.e.   1.6299   1.7001  5.1929   6.1762  3.5431  0.8653`
	
	`sigma^2 estimated as 13.94:  log likelihood = -131.28,  aic = 274.56`

ARIMA(2,2,1)	
	`> (fit4 <- arima(ts_diff, order = c(2,2,1)))`
	
	`Call:`
	`arima(x = ts_diff, order = c(2, 2, 1))`
	
	`Coefficients:`
	`          ar1      ar2      ma1`
	`      -1.2384  -0.6923  -1.0000`
	`s.e.   0.1009   0.0985   0.0601`
	
	`sigma^2 estimated as 24.54:  log likelihood = -139.87,  aic = 285.74`

ARIMA(2,2,4)	
	`> (fit5 <- arima(ts_diff, order = c(2,2,4)))`
	
	`Call:`
	`arima(x = ts_diff, order = c(2, 2, 4))`
	
	`Coefficients:`
	`          ar1      ar2      ma1     ma2     ma3      ma4`
	`      -1.2361  -0.5367  -1.7490  0.1636  0.9268  -0.3396`
	`s.e.   0.2311   0.1418   0.3913  0.7066  0.6598   0.2716`
	
	`sigma^2 estimated as 12.94:  log likelihood = -130.39,  aic = 272.79`

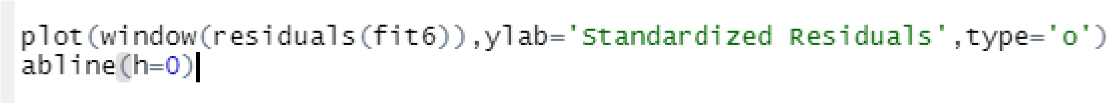
ARIMA(2,2,5)



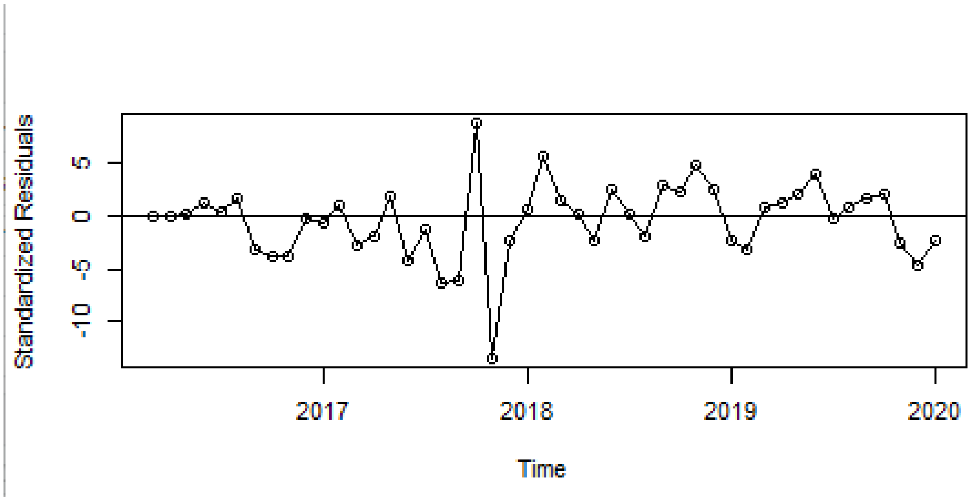
```
> (fit6 <- arima(ts_diff, order = c(2,2,5)))

Call:
arima(x = ts_diff, order = c(2, 2, 5))

Coefficients:
          ar1      ar2      ma1     ma2     ma3      ma4     ma5
      -1.0150  -0.2931  -1.9884  0.6099  1.0188  -0.9012  0.2641
s.e.   0.4505   0.4386   0.4843  0.9346  0.4924   0.8726  0.4099

sigma^2 estimated as 13.1:  log likelihood = -130.23,  aic = 274.47
>
```

Based on the different models , we can see that ARIMA(2,2,5) had the least AIC value, sigma^2 being the least therefore is the best model for given time series. Find the below time series plot for the residuals.

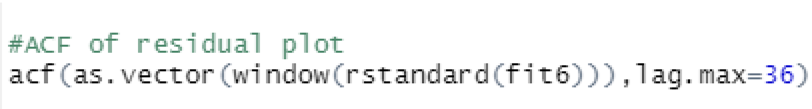
5.8 Residual Analysis:



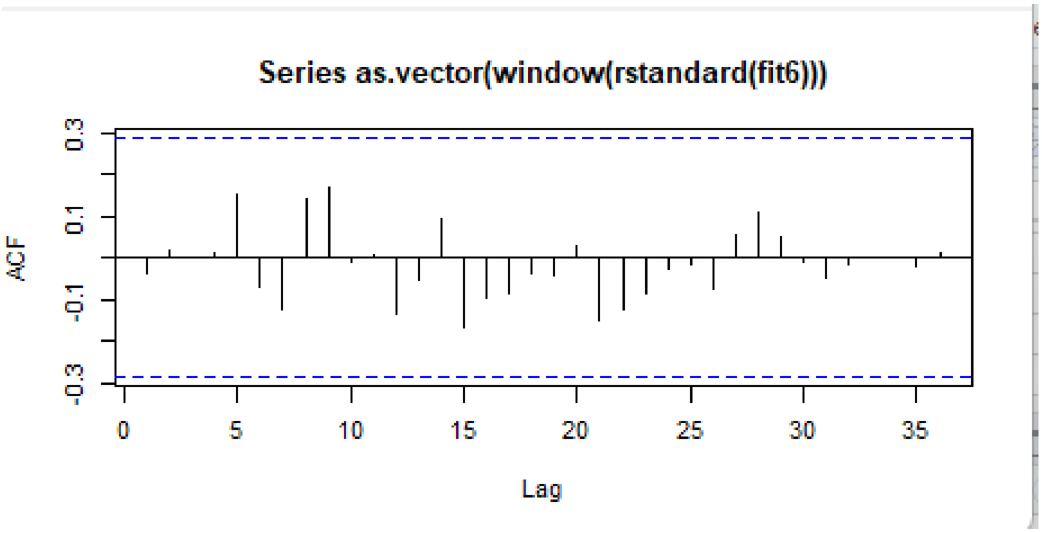
```
plot(window(residuals(fit6)),ylab='Standardized Residuals',type='o')
abline(h=0)
```



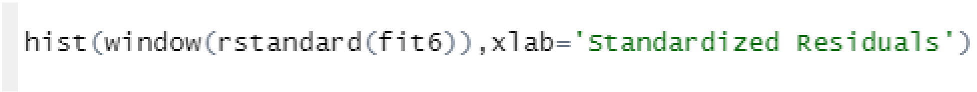
Residual plot tells the points that are left after fitting the model. We can see that most points are closer to the line except at the middle of the plot. Now lets plot the ACF of residuals for the model to further understand its behavior.



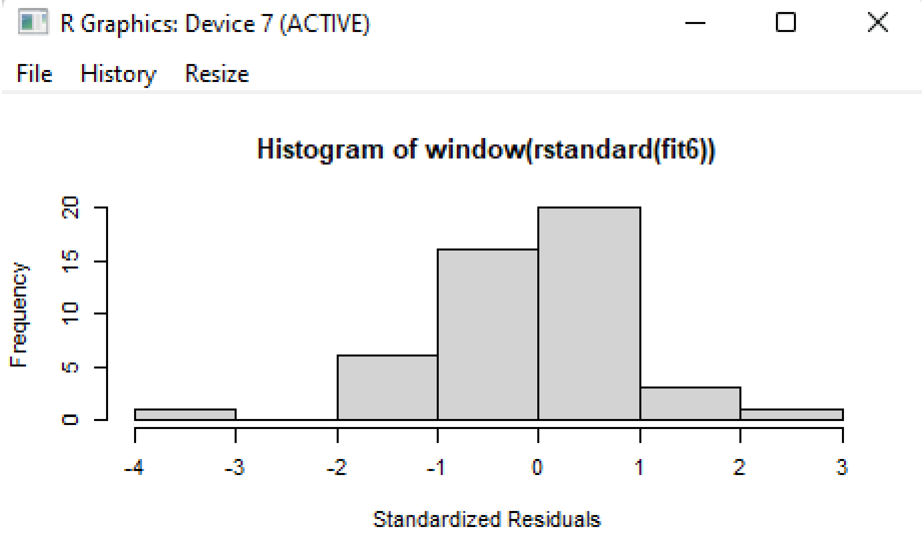
```
#ACF of residual plot
acf(as.vector(window(rstandard(fit6))),lag.max=36)
```



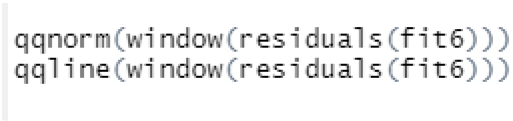
From the plot for ACF of residuals ,we can clearly see that there is no statistically significant correlation for the data and every point is within the confidence interval.



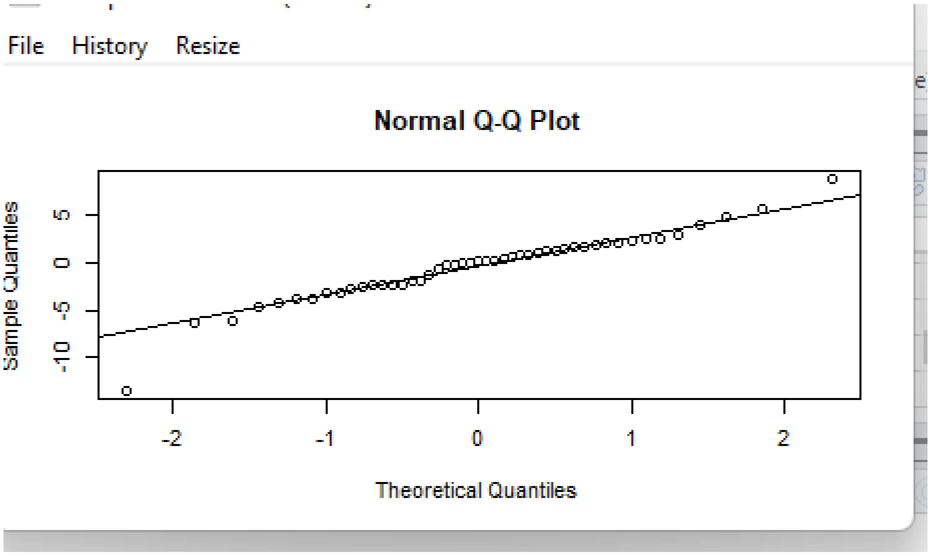
```
hist(window(rstandard(fit6)),xlab='Standardized Residuals')
```



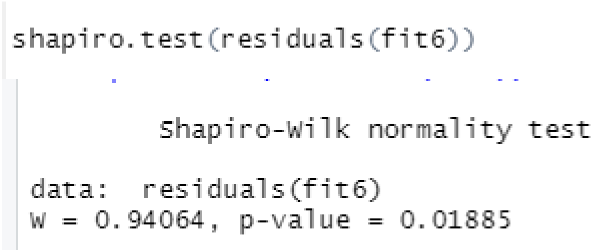
From the histogram we can see that , it slightly follow normal distribution if we ignore the outliers. But the plot is slightly right skewed in nature .For more understanding we need to perform quantile-quantile plot for the analysis.



```
qqnorm(window(residuals(fit6)))
qqline(window(residuals(fit6)))
```



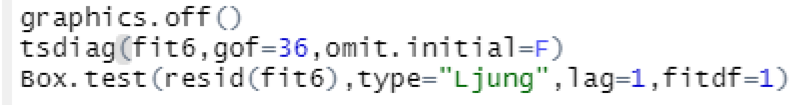
From the qqplot for the residuals we can say that, most of the points lie on the reference line, however they are few points towards the tail part of the plots that deviate slightly . QQ plot gives a better visual of the residuals how the sample quantiles are related to the theoretical quantiles.There are few tests which can be performed , to check the normality of the residuals one such is Shapiro test.



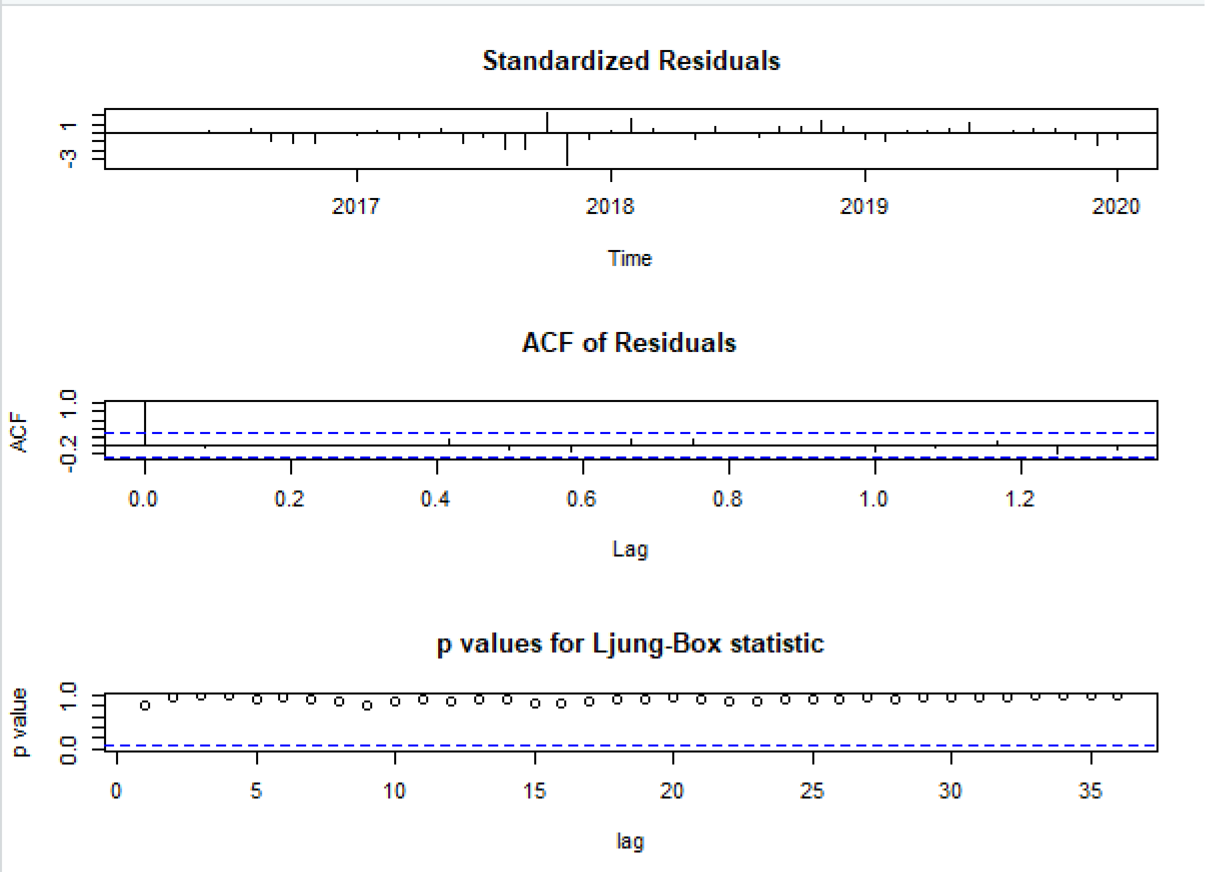
```
shapiro.test(residuals(fit6))

        Shapiro-Wilk normality test

data:  residuals(fit6)
W = 0.94064, p-value = 0.01885
```

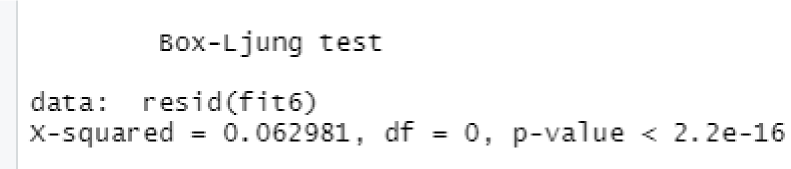
The above figure shows the results of Shapiro-wilk test for the residuals of the model. If the value of p is equal to or less than 0.05, then the hypothesis of normality will be rejected by the Shapiro test. Here the p value is less than 0.05 so we can say that the residuals follow normal distribution.

Ljung-Box test is next performed on the models to test the randomness of the data over the lags at the bigger perspective.



```
graphics.off()
tsdiag(fit6,gof=36,omit.initial=F)
Box.test(resid(fit6),type="Ljung",lag=1,fitdf=1)
```

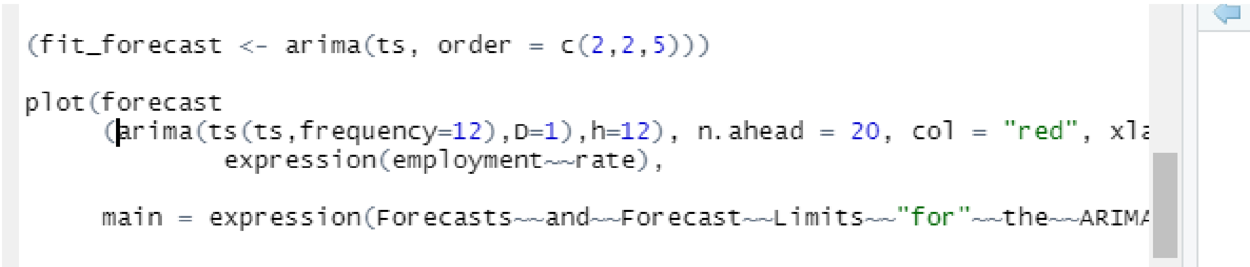




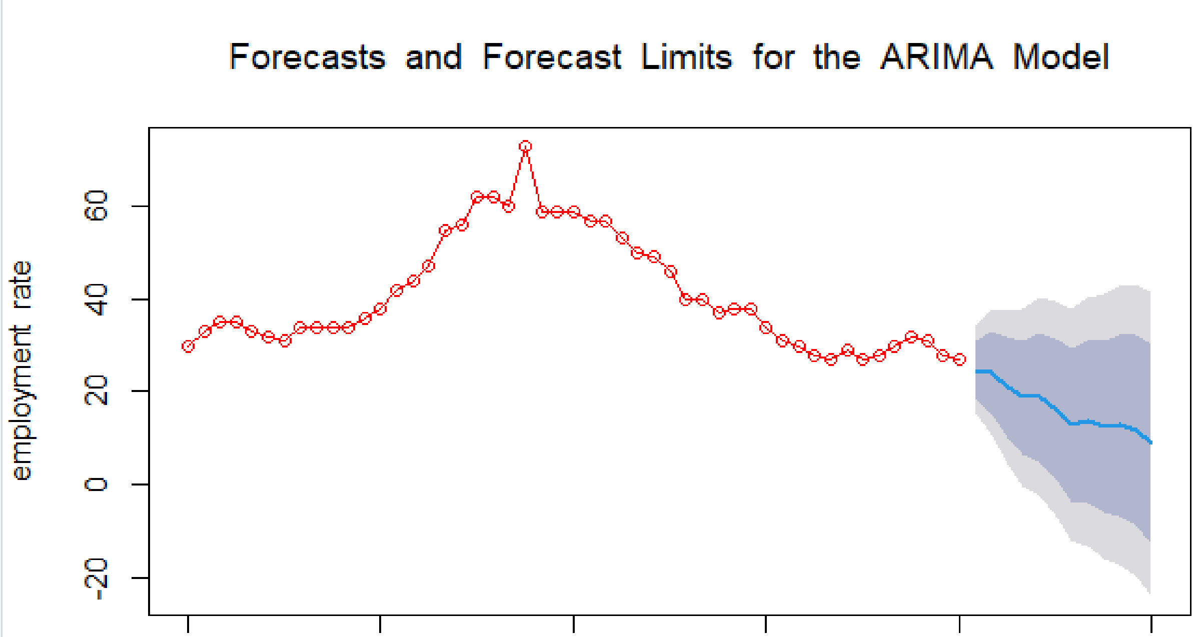
```
        Box-Ljung test

data:  resid(fit6)
X-squared = 0.062981, df = 0, p-value < 2.2e-16
```

The null hypothesis for LB test is that residuals are independently distributed if p values is less than 0.05. Based on that we can see that , independence has been captured by the data for the following model.

5.9 Time series forecasting:



```
(fit_forecast <- arima(ts, order = c(2,2,5)))

plot(forecast
     (arima(ts(ts,frequency=12),D=1),h=12), n.ahead = 20, col = "red", xla
               expression(employment~~rate),

     main = expression(Forecasts~~and~~Forecast~~Limits~~"for"~~the~~ARIMA
```



The plot shows the forecasting to plot for the next 20 values which is shown by the blue region.

6. Executive Summary :

The report talks about the step by step procedure to perform time series analysis and forecasting using various methodologies. The report consists of 2 datasets one being seasonal and the other non-seasonal where data is being loaded in R and uses different libraries like TSA, tseries, forecast, rugarch to perform the analysis . Major part of the report discusses about how ACF and PACF plot are very much important in deducing the parameters for the ARIMA model. Stationary check is mandatory to perform any time series analysis. The time series analysis to find seasonality and so on , need clear understanding of the significance of 3 kinds of plots namely acf , pacf and time series plot. Further on fitting different models by varying the p , q and differentiation values we can parameters like sigma^2 , logliklihood and AIC value which need to be analyzed . It has been formulated in finding the best module the AIC value has to be the smallest and the log likelihood has to be higher . Further various residual techniques are done on the best model to find does it satisfy normality or autocorrelation and histogram analysis to clearly see if the model forms normal distribution or is skewed in nature. The qq plot is one of the techniques used which was quite useful to know the behavior of outliers for the models. Further the last section we perform forecasting on the original time series to see how it can find further values based on the mentioned time period say 20 months or so on.

7. Challenges faced and conclusion:

I would like to thank the time series Professor to provide me with clear roadmap to solve any time series dataset . His step by step procedure and algorithm really helped me to solve both of this datasets. Some of the challenges I faced was during preprocessing of the Seasonal dataset which was in a different format and getting it to time series format . Because there was different ways to get it to time series format like does it have to be monthly data or yearly data or which category to choose form and so on . Finding Seasonal trends was another challenge as ACF plot had pins not very prominent and thus I resampled the time series data into yearly to see monthly patterns for multiple years and thus understood the seasonality than consider the entire dataset at once and analyzing it . Model selection and fitting the model was another challenge , but I was able to do it based on clear understanding of ACF and PACF plots. Understanding various Residual methodologies required clear understanding of the Statistical methods , so I had to read in deep about these techniques and analyze for the dataset.