TIME SERIES ANALYSIS

ARIMA MODEL IN R

Stock Price and Exchange Rate

TABLE OF CONTENTS

Page no.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our supervisor, Professor **Dr Khanindra Ch. Das**, for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in our research and writing of this assignment. His immense knowledge, profound experience and professional expertise in Data Quality Control have enabled us to complete this assignment named **"Time series Analysis - ARIMA model in R"** successfully.

Without his support and guidance, this assignment would not have been possible. We could not have imagined having a better supervisor in our study.

EXECUTIVE SUMMARY

Time series is a series of data points in which each data point is associated with a timestamp. A simple example is a price of a stock in the stock market at different points of time on a given day. Another example is the Exchange Rate. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called a time-series object. It is also an R data object like a vector or data frame.

ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

A key role in time series analysis is played by processes whose properties, or some of them, do not vary over time. Such a property is illustrated in the following important concept, stationarity. We then introduce the most commonly used stationary linear time series models–the autoregressive integrated moving average (ARIMA) models. These models have assumed great importance in modelling real-world processes.

ARIMA MODEL IN R

Data File – Exchange Rate, US Dollar

(You can download from this Link

https://dbie.rbi.org.in/DBIE/dbie.rbi?site=home)

	A	B
1	**Date**	**US Dollar**
2	01-01-2020	71.37
3	02-01-2020	71.34
4	03-01-2020	71.69
5	06-01-2020	72.09
6	07-01-2020	71.78
7	08-01-2020	72.02
8	09-01-2020	71.42
9	10-01-2020	71.11
10	13-01-2020	70.81
11	14-01-2020	70.92
12	15-01-2020	70.88
13	16-01-2020	70.91
14	17-01-2020	71.04
15	20-01-2020	71.06
16	21-01-2020	71.18
17	22-01-2020	71.21

Total No. of Data - 460

(Sample of the file and we give the name – ExRate)

We have chosen past data from **01-01-2020** to **29-07-2020**, to see how covid-19 impact the exchange rate (US Dollar) to Indian Economy how we will see the exchange rate upcoming future, so that we need to do forecasting in R where we will use ARIMA model to see the changes and to predict the Exchange Rate.

1st you convert the Excel file to CSV file, so that it is easy to import in R.

Next you should import the file in to R. Before that you should select the path, for path selection you should press CTRL + SHIFT + H, it will be asked the folder where your data file already exist. You should select the folder.

Here you can see that our syntax or path –

Syntax- setwd

setwd("D:\CLASSES\4TH TRIMESTAR\BUSINESS VERTICAL\TIME SERIES\ASSIGNMENT\US DOLLAR EXCHANGE RATE")

After that you should declare your data by read.csv

Syntax – read.csv

Exchange_Rate<-read.csv("ExRate.csv")

If you want to see your data file then you can use view command to see the data.

Syntax - View

View(Exchange_Rate)

Exchange Rate.R* Exchange_Rate

Filter

	Date	US.Dollar
1	2020-01-01	71.37
2	2020-01-02	71.34
3	2020-01-03	71.69
4	2020-01-06	72.09
5	2020-01-07	71.78
6	2020-01-08	72.02
7	2020-01-09	71.42
8	2020-01-10	71.11
9	2020-01-13	70.81
10	2020-01-14	70.92
11	2020-01-15	70.88
12	2020-01-16	70.91
13	2020-01-17	71.04
14	2020-01-20	71.06

You can use head function for seeing 1st 6 data from your data file.

Syntax - head

head(Exchange_Rate)

```
> head(Exchange_Rate)
        Date US.Dollar
1 2020-01-01     71.37
2 2020-01-02     71.34
3 2020-01-03     71.69
4 2020-01-06     72.09
5 2020-01-07     71.78
6 2020-01-08     72.02
```

You can use tail function to see last 6 data from your data file.

Syntax – tail

tail(Exchange_Rate)

```
> tail(Exchange_Rate)
          Date US.Dollar
455 2022-07-22     79.91
456 2022-07-25     79.85
457 2022-07-26     79.79
458 2022-07-27     79.90
459 2022-07-28     79.74
460 2022-07-29     79.42
```

If you want summary of the data then you can use summary function.

Syntax- summary

summary(Exchange_Rate)

```
> summary(Exchange_Rate)
     Date             US.Dollar    
 Length:460         Min.   :70.81  
 Class :character   1st Qu.:73.15  
 Mode  :character   Median :74.44  
                    Mean   :74.67  
                    3rd Qu.:75.83  
                    Max.   :79.98  
```

Before we proceed further, we should know the structure of our data set. For this we use str function to know the structure.

Syntax – str

str(Exchange_Rate)

```
> str(Exchange_Rate)
'data.frame':	460 obs. of  2 variables:
 $ Date     : chr  "2020-01-01" "2020-01-02" "2020-01-03" "2020-01-06" ...
 $ US.Dollar: num  71.4 71.3 71.7 72.1 71.8 ...
```

Here it is seen that our date is in the character form and the US. Dollar is numerical form.

So, for time series we need to convert of date to Date format so that we can proceed further. For this we will use as.Date function and declare a variable rdate.

Syntax – as.Date

rdate<-as.Date(Exchange_Rate$Date)

Now we can fix this variable because further it will be not changed.

Syntax- fix

fix(rdate)

Now you can check the structure of the our variable that we declared rdate where the date of the our data set exists.

Syntax – str

str(rdate)

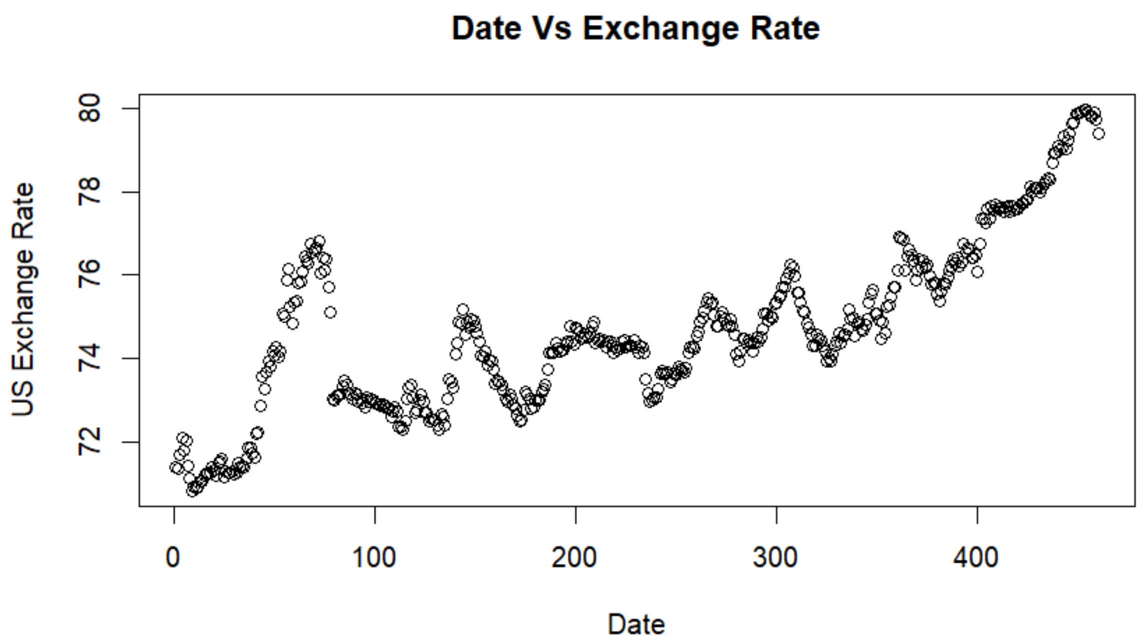
```
> str(rdate)
 Date[1:460], format: "2020-01-01" "2020-01-02" "2020-01-03" "2020-01-06" "2020-01-07" "2020-01-08"
 ...
```

Now you can plot graph **Date Vs Exchange Rate**

Syntax – plot

plot(Exchange_Rate$US.Dollar,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate")

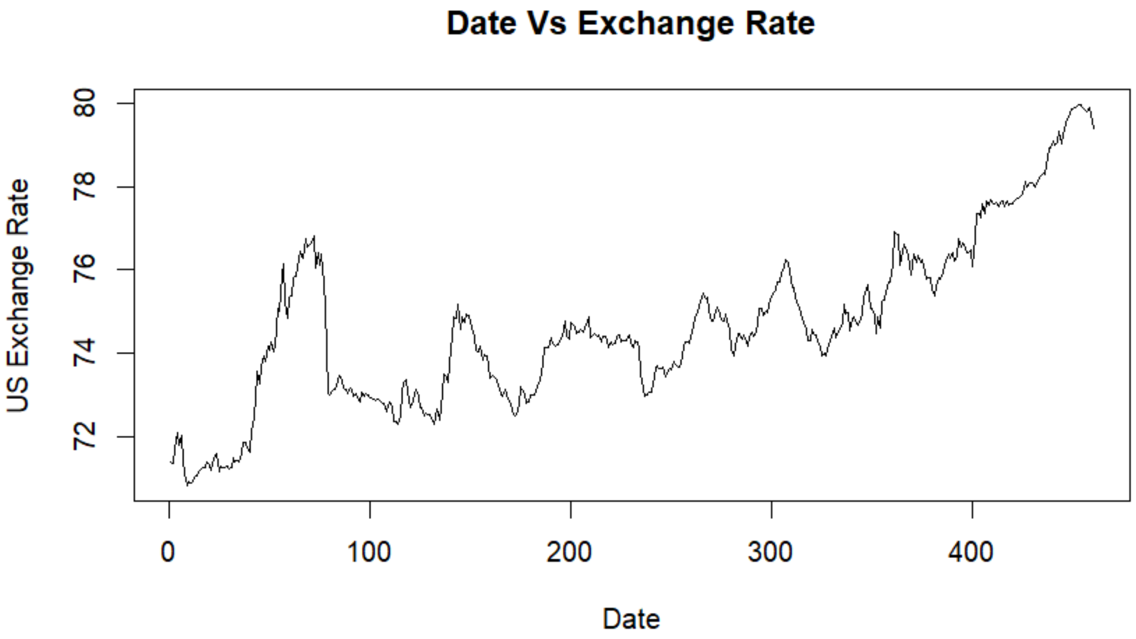
Note **-** $ sign we use for we want US.Dollar from our data set Exchange_Rate for plot against Date which is x axis, xlab & ylab used for labelling x axis and y axis "Date" and "Us Exchange Rate" respectably, main function we used for what we want to give heading of our graph, here the heading is "Date Vs Exchange Rate".



Now you can plot time series graph for better understanding same Date Vs Exchange Rate which we have drawn before.

Syntax – plot.ts

```
plot.ts(Exchange_Rate$US.Dollar,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate")
```



For better visualization we can draw ggplot which gives us better understanding and better visualization effect. Before that we should import ggplot2 and scales for drawing ggplot.

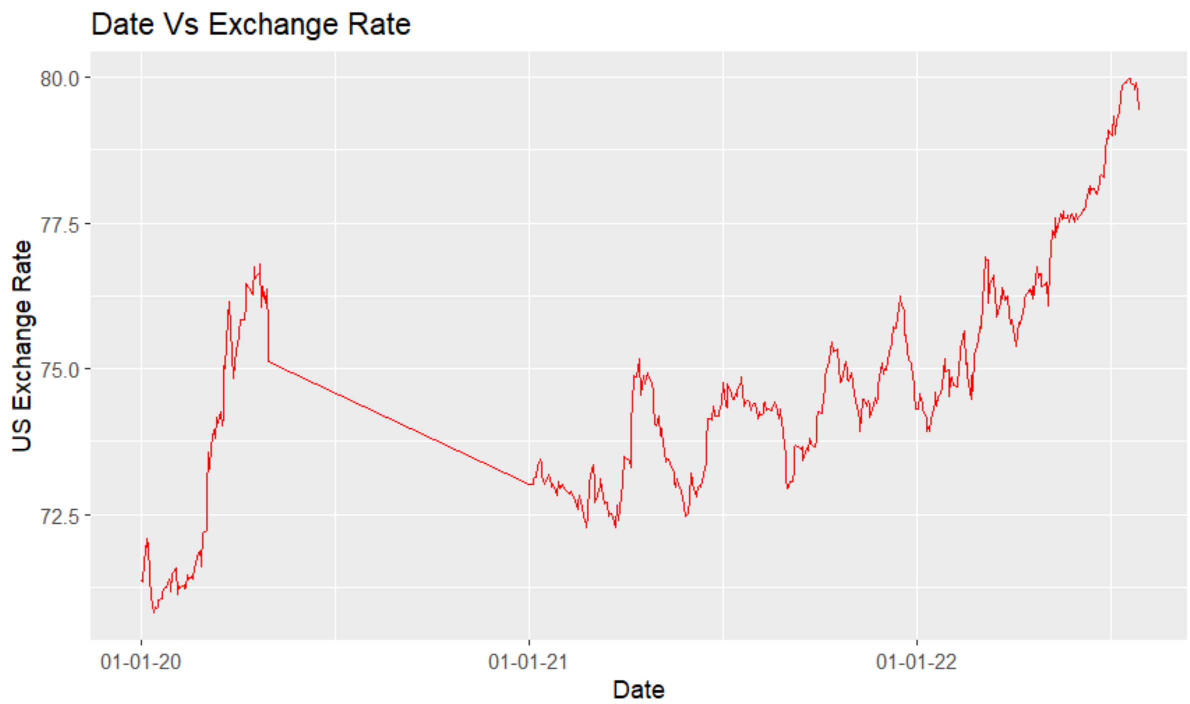
```
library(ggplot2)

library(scales)
```

Syntax – ggplot

```
ggplot(data=Exchange_Rate,aes(x=rdate,y=US.Dollar))+geom_line(color="red")+scale_x_date(labels=date_format("%d-%m-%y"))+labs(x="Date",y="US Exchange Rate",title="Date Vs Exchange Rate")
```

Note – data is of our data set, aes denotes what is our x-axes and y-axes to be, geom_line denotes highlighting exactly when changes occur. Here we write in x-axes how we want our date format like (d-m-y) or any format you should mention here and rest labs and title will be same as our previous graphs. Colour also you mention if you want.

Here is our graph –



Now, it's time to convert our data set in to time series format. This step is very important for time series because without this step we can't proceed our prediction which is our goal. For our prediction we need our data set in time series format.

Before that we should import xts function and declare library.

library(xts)

xts function is known Constructor function for creating an extensible time-series object.

Syntax – xts

Exchangepricetime=xts(Exchange_Rate$US.Dollar,rdate)

We gave another name Exchangepricetime where we convert both rdate and US.Dollar both into time series format.

Now if you want to see the structure of our new data set Exchangepricetime then use the function str.

Syntax – str

str(Exchangepricetime)

```
> str(Exchangepricetime)
An 'xts' object on 2020-01-01/2022-07-29 containing:
  Data: num [1:460, 1] 71.4 71.3 71.7 72.1 71.8 ...
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
 NULL
```

For better understand you can use class function also –

Syntax- class

class(Exchangepricetime)

```
> class(Exchangepricetime)
[1] "xts" "zoo"
```

Now you can see our data file is converted into time series format and the name of our new data set is Exchangepricetime.

Now come to forecast session which is our objective

Before we forecast, we should import tseries and forecast package from our library.

library(forecast)

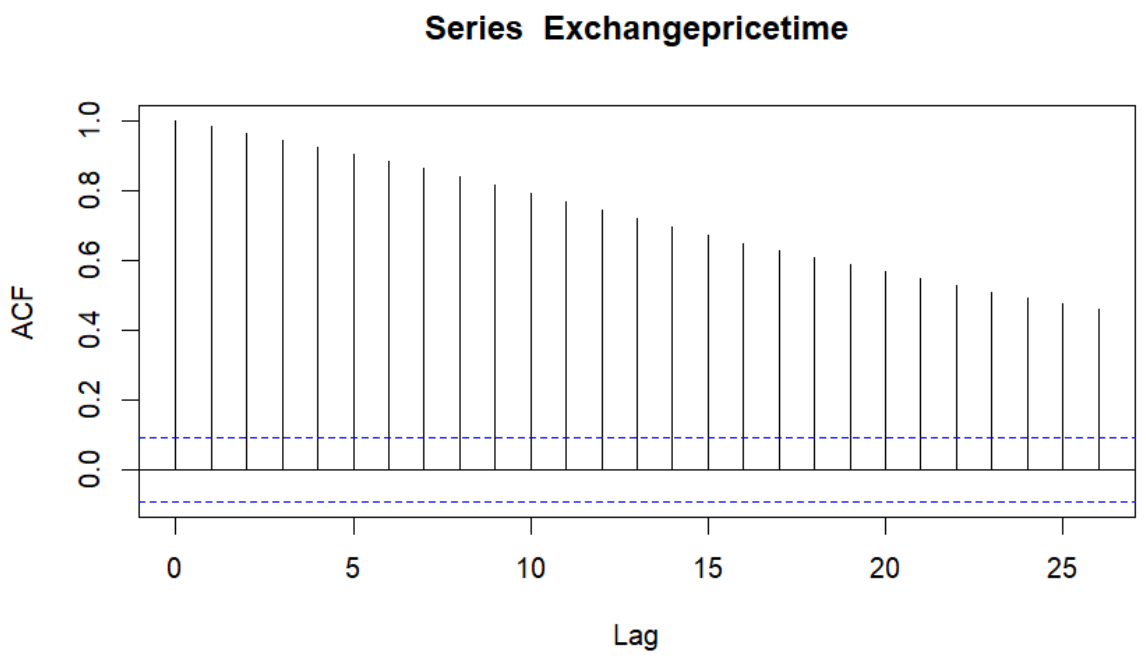
library(tseries)

Before we go further we should know either our data is stationary or not for that we can check acf, pacf and adf test.

acf test (Autocorrelation function)-

Syntax – acf

acf(Exchangepricetime)



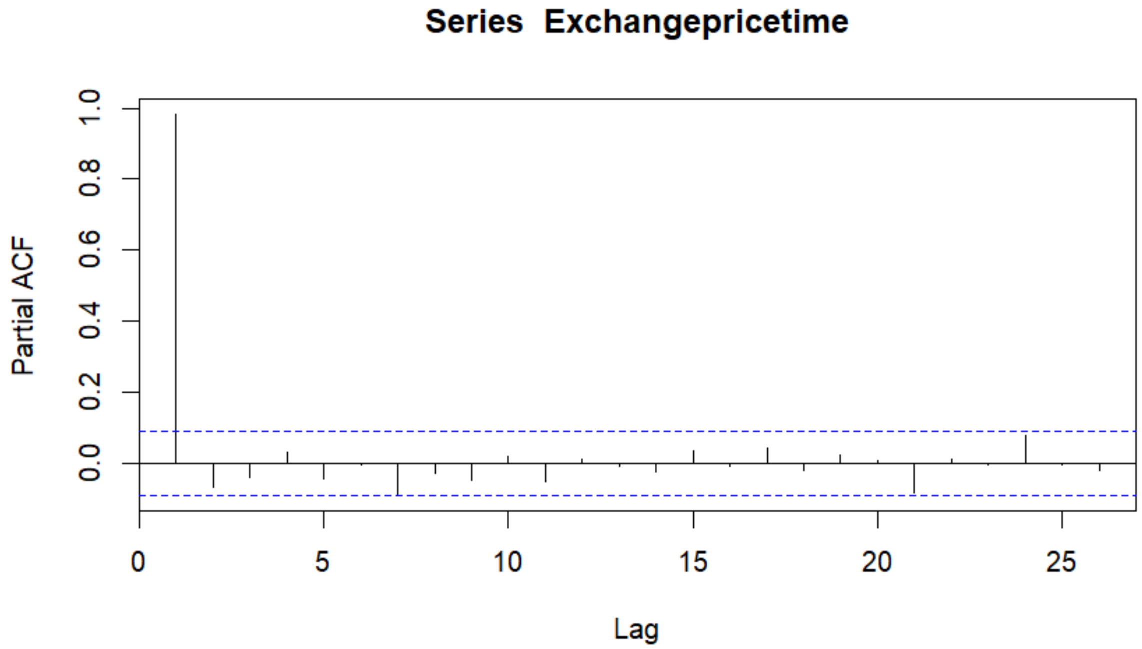
The two blue lines are called significance levels and the interpretation is for stationarity all the lines come should come under these two blue lines but here all the lines are above the blue lines. So, the data is not stationary.

We can also test pacf to know stationary –

Syntax- pacf

pacf = (Partial Autocorrelation Function)

pacf(Exchangepricetime)



Here also we drawn the same interpretation.

Now we can test the adf

adf test (Augmented Dickey - Fuller test)

Syntax – adf.test

adf.test(Exchangepricetime)

```
> adf.test(Exchangepricetime)

        Augmented Dickey-Fuller Test

data:  Exchangepricetime
Dickey-Fuller = -2.838, Lag order = 7, p-value = 0.2235
alternative hypothesis: stationary
```

Interpretation – if p value is greater than 0.05, we fail to reject the null hypothesis that means data is not stationary.

Now we can test our ARIMA model

Syntax- arima

model1<-arima(Exchangepricetime,order = c(0,1,0))

```
> model1

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


sigma^2 estimated as 0.06547:  log likelihood = -25.64,  aic = 53.27
```

model2<-arima(Exchangepricetime,order = c(1,1,1))

```
> model2

Call:
arima(x = Exchangepricetime, order = c(1, 1, 1))

Coefficients:
         ar1      ma1
      0.7910  -0.7298
s.e.  0.1866   0.2090

sigma^2 estimated as 0.06482:  log likelihood = -23.36,  aic = 52.73
```

model3<-arima(Exchangepricetime,order = c(0,1,1))

```
> model3

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

Coefficients:
         ma1
      0.0624
s.e.  0.0431

sigma^2 estimated as 0.06517:  log likelihood = -24.59,  aic = 53.19
```

model4<-arima(Exchangepricetime,order = c(1,1,0))

```
> model4

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

Coefficients:
         ar1
      0.0725
s.e.  0.0466

sigma^2 estimated as 0.06512:  log likelihood = -24.43,  aic = 52.85
```

OR

also, we check though auto arima function. Which will give us best arima model for our data.

Syntax – auto.arima

model=auto.arima(Exchangepricetime,ic="aic",trace = TRUE)

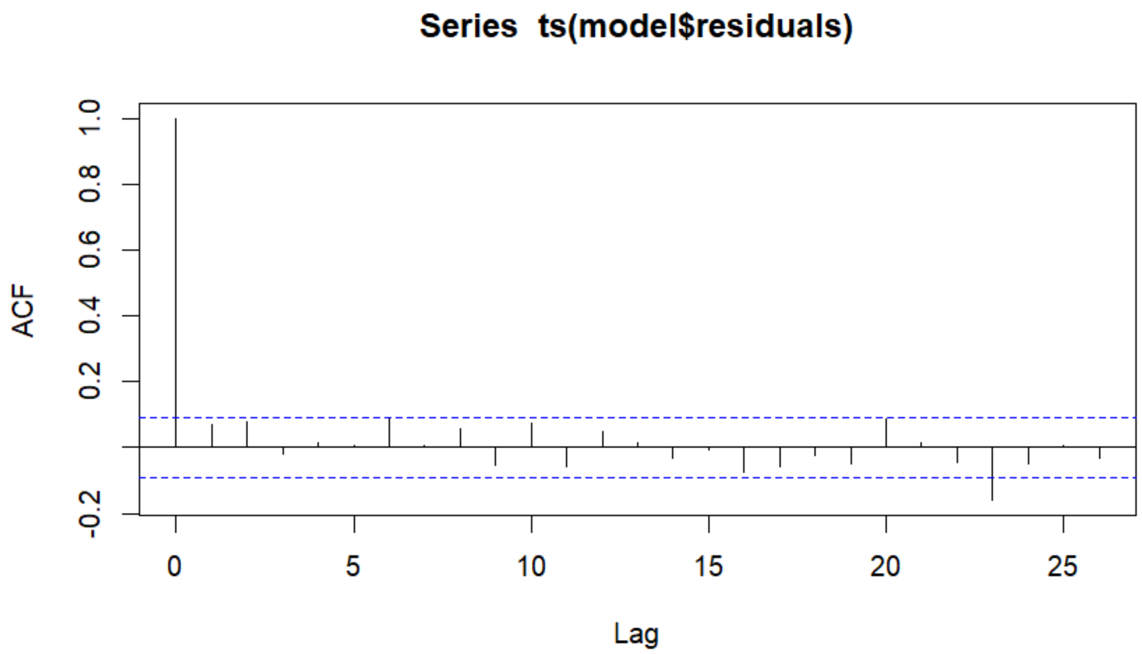
```
Fitting models using approximations to speed things up...

ARIMA(2,1,2) with drift         : 58.36751
ARIMA(0,1,0) with drift         : 55.8347
ARIMA(1,1,0) with drift         : 56.67563
ARIMA(0,1,1) with drift         : 55.99293
ARIMA(0,1,0)                    : 55.99625
ARIMA(1,1,1) with drift         : 57.36565

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

ARIMA(0,1,0) with drift         : 53.10852


 Best model: ARIMA(0,1,0) with drift
```

To know best model, we should check the AIC value which is less is known for best model

Now we can also check acf and pacf for stationarity

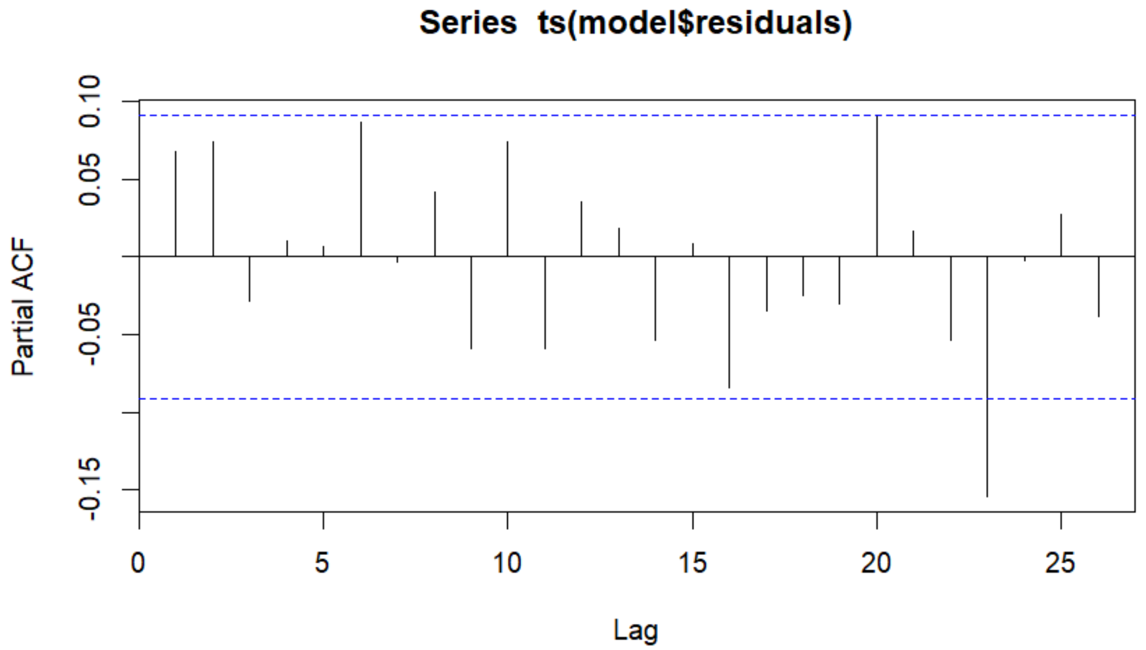
Syntax - acf

acf(ts(model$residuals))



Syntax – pacf

pacf(ts(model$residuals))



Now you can see all the lines with in significance level between blue line. Now you can say that data is stationary.

The final step is our forecasting –

Syntax – forecast

myexchangerateforecast=forecast(model,level = c(95),h=10)

with 95 confidence interval and we want to forecast next 10 days US Dollar Exchange Rate.

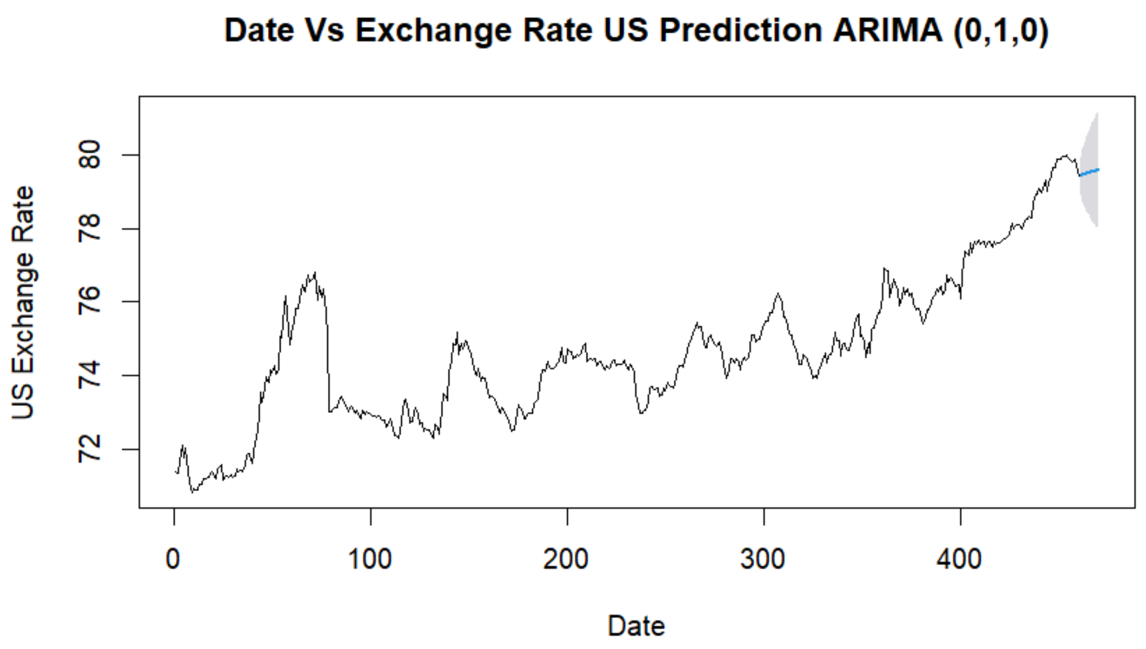
```
> myexchangerateforecast
    Point Forecast    Lo 95    Hi 95
461       79.43754 78.93663 79.93844
462       79.45508 78.74669 80.16346
463       79.47261 78.60502 80.34020
464       79.49015 78.48835 80.49196
465       79.50769 78.38764 80.62775
466       79.52523 78.29827 80.75219
467       79.54277 78.21750 80.86803
468       79.56031 78.14354 80.97707
469       79.57784 78.07513 81.08055
470       79.59538 78.01139 81.17938
```

Now we can forecast our model through graph –

Using plot function

Syntax – plot

plot(myexchangerateforecast,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate US Prediction ARIMA (0,1,0)")



Now we check the accuracy of our model –

Syntax – accuracy

accuracy(myexchangerateforecast)

```
> accuracy(myexchangerateforecast)
                      ME      RMSE       MAE           MPE      MAPE      MASE       ACF1
Training set 0.000155114 0.2550115 0.1745735 -0.0005868246 0.2339544 0.9947764 0.06771125
```

Where

ME = Mean Error

RMSE = Root Mean Squared Error

MAE = Mean Absolute Error

MPE = Mean Percentage Error

MAPE = Mean Absolute Percentage Error

MASE = Mean Absolute Scaled Error

Accuracy of our Model = 100 – MAPE = 99.76605

The performance of a forecasting model should be the baseline for determining whether your values are **good**. It is irresponsible to set arbitrary **forecasting** performance targets (such as **MAPE** < 10% is Excellent, **MAPE** < 20% is **Good**) without the context of the forecast ability of your data.

Root Mean Square Error (**RMSE**) is the standard deviation of the residuals (prediction errors). ... **Root mean square error** is commonly used in climatology, **forecasting**, and regression analysis to verify experimental results.

MAPE stands for Mean Absolute Percent Error **- Bias** refers to persistent **forecast** error **- Bias** is a component of total calculated **forecast** error **- Bias** refers to consistent under-**forecasting** or over-**forecasting** **- MAPE** can be misinterpreted and miscalculated, so use caution in the interpretation.

Using this **RMSE** value, according to NDEP (National Digital Elevation Guidelines) and FEMA guidelines, a measure of **accuracy** can be computed: **Accuracy** = 1.96***RMSE**. This **Accuracy** is stated as: "The fundamental vertical **accuracy** is the value by which vertical **accuracy** can be equitably assessed and compared among datasets.

The similarity between them is they both measure the absolute error. So in both the negative and the positive errors, cancel out each other. **RMSE** method is more accurate. ... The difference between them is, **MAPE** measures the deviation from the actual data in terms of percentage, that is the only difference between them.

It means that there is no absolute **good** or bad threshold, however you can define it based on your DV. For a datum which ranges from 0 to 1000, an **RMSE** of 0.7 is small, but if the range goes from 0 to 1, it is not that small anymore.

The **RMSE** is the square root of the variance of the residuals. ... **Lower** values of **RMSE** indicate **better** fit. **RMSE** is a **good** measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

The **RMSE is** a quadratic scoring rule which measures the average magnitude of the error. ... Since the errors **are** squared before they **are** averaged, the **RMSE** gives a relatively high weight to large errors. This means the **RMSE is** most useful when large errors **are** particularly undesirable.

Try to play with other input variables, and compare your **RMSE** values. The smaller the **RMSE** value, the better the model. Also, try to compare your **RMSE** values of both training and testing data. If they are almost similar, your model is good.

The **RMSE** result will always be larger or equal **to** the **MAE**. If all **of** the errors have the same magnitude, then **RMSE**=**MAE**. [**RMSE**] ≤ [**MAE** * sqrt(n)], where n is the number **of** test samples. The difference **between** **RMSE** and **MAE** is greatest when all **of** the prediction error comes from a single test sample.

What is the meaning of residuals in time series forecasting analysis?

Let's take u want to predict the value of a series at the next time instant. That means you're interested in finding one step ahead prediction value of a given series. When you compare this predicted value with the observed one whatever difference you get is called Residual. See, you will always find some difference and now the key is to analyse these residuals. For e.g., by looking at the autocorrelation of the residuals you will try to find if any relation exists between them. If not then you can say you have predicted well enough. If not, then your model is not good enough to capture the series behaviour. So go and check your model structure or order.

In short, Residual analysis suggests you about your estimation model quality.

Now we want residual for test normality graph – (from our best fit Model)

The power of **Q-Q plots** lies in their ability to summarize any distribution visually.

QQ plots is very useful to determine

- If two populations are of the same distribution
- If residuals follow a normal distribution. Having a normal error term is an assumption in regression and we can verify if it's met using this.
- Skewness of distribution

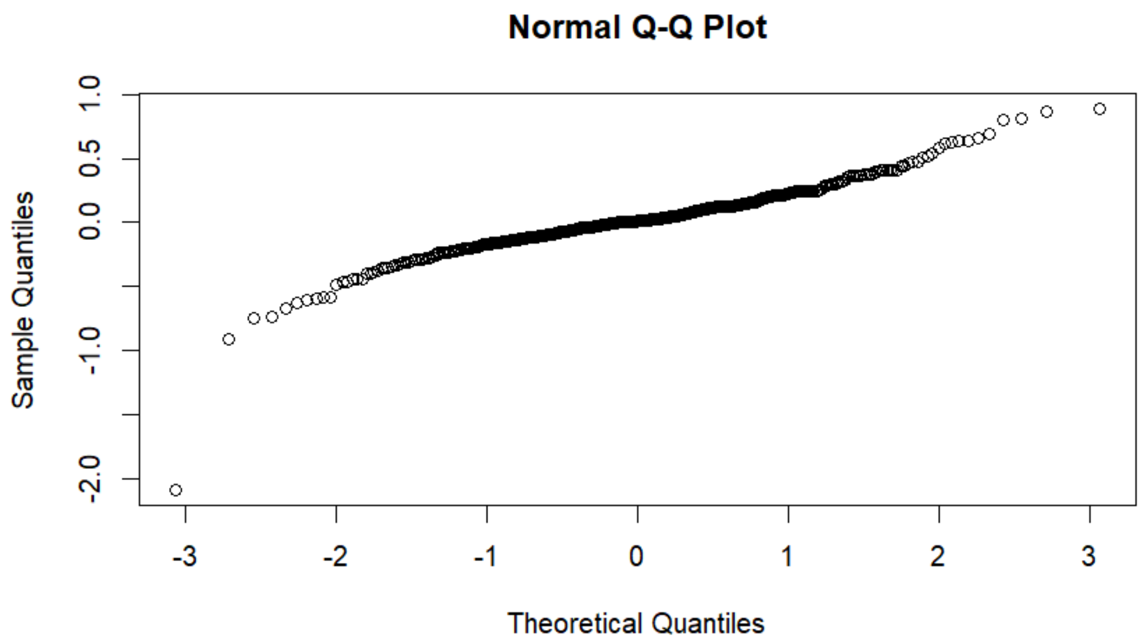
Syntax - residuals

residual<-residuals(model1)

Now we draw the graph

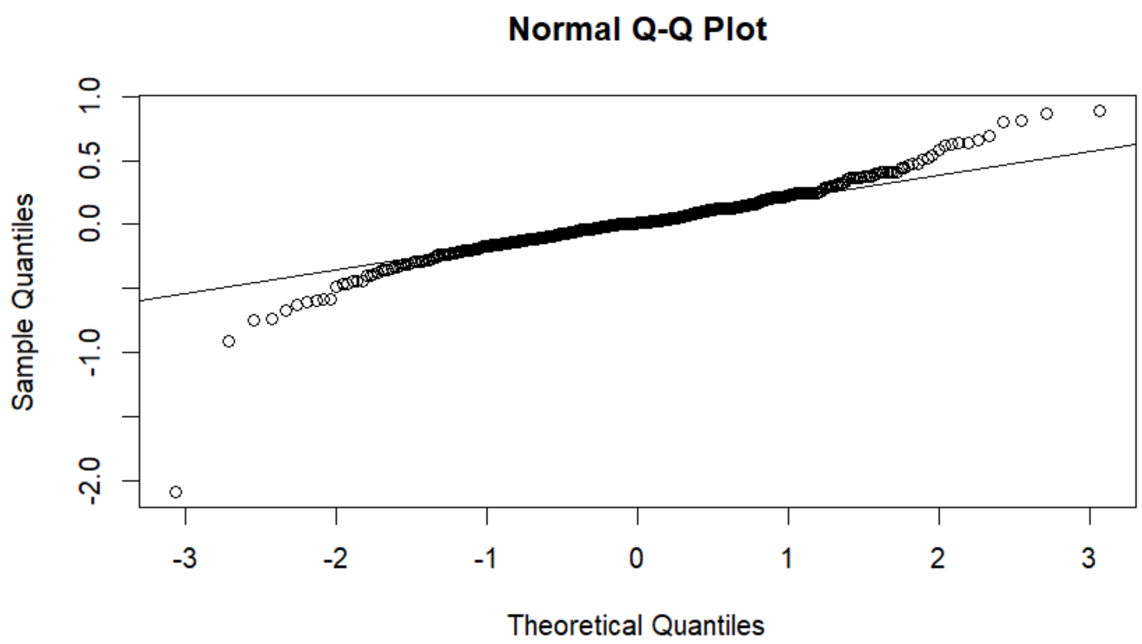
Syntax – qqnorm

qqnorm(residual)



Syntax - qqline

qqline(residual)



The Quantiles-Quantiles plot (Q-Q Plot) is a qualitative way of assessing whether or not sample data could possibly have been drawn from some distribution typically normal distribution.

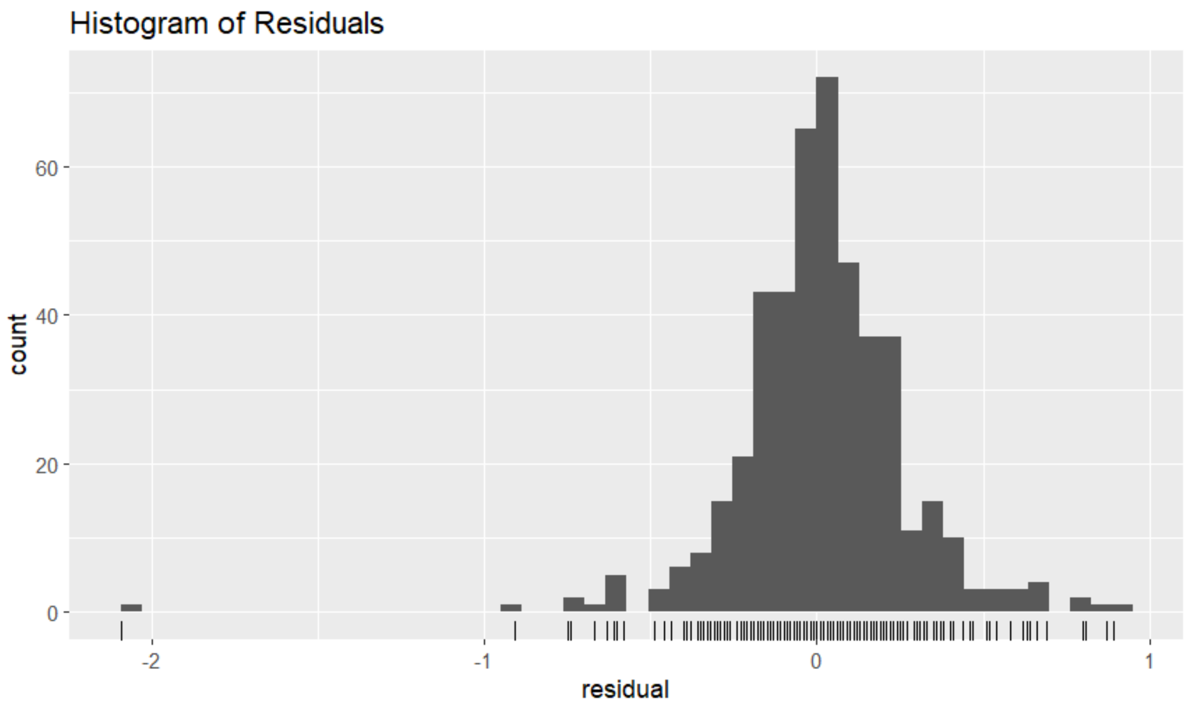
Here our distribution is symmetric with fat tails.

In Q-Q plots, we plot the theoretical Quantile values with the sample Quantile values. Quantiles are obtained by sorting the data. It determines how many values in a distribution are above or below a certain limit.

Also, we can see that histogram of residuals –

Syntax – gghistogram

gghistogram(residual)+ggtitle("Histogram of Residuals")



Here we can say that fat tails of our residuals in normality graph.

Next, we see Ljung-Box Test –

Syntax – Box.test

Box.test(myexchangerateforecast$resid, lag=5, type= "Ljung-Box")

```
> Box.test(myexchangerateforecast$resid, lag=15, type= "Ljung-Box")

        Box-Ljung test

data:  myexchangerateforecast$resid
X-squared = 17.503, df = 15, p-value = 0.2897
```

The Ljung-Box test uses the following hypotheses:

H_0: The residuals are independently distributed.

H_A: The residuals are not independently distributed; they exhibit serial correlation.

Ideally, we would like to fail to reject the null hypothesis. That is, we would like to see the p-value of the test be greater than 0.05 because this means the residuals for our time series model are independent, which is often an assumption we make when creating a model.

The test statistic of the test is Q = **5.1878** and the p-value of the test is **0.3934**, which is much larger than 0.05. Thus, we fail to reject the null hypothesis of the test and conclude that the data values are independent.

Note that we used a lag value of 05 in this example, but you can choose any value that you would like to use for the lag, depending on your particular situation.

Like –

Box.test(myexchangerateforecast$resid, lag=15, type= "Ljung-Box")

```
> Box.test(myexchangerateforecast$resid, lag=15, type= "Ljung-Box")

        Box-Ljung test

data:  myexchangerateforecast$resid
X-squared = 17.503, df = 15, p-value = 0.2897
```

Other tests also we can do for measuring the stationarity like – adf test and jarque.bera.test

Syntax – adf.test

adf.test(myexchangerateforecast$resid)

```
> adf.test(myexchangerateforecast$resid)

        Augmented Dickey-Fuller Test

data:  myexchangerateforecast$resid
Dickey-Fuller = -6.4367, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(myexchangerateforecast$resid) :
  p-value smaller than printed p-value
```

P value smaller than 0.05 so, the data is stationary.

Similarly –

Syntax - jarque.bera.test

jarque.bera.test(residual)

```
> jarque.bera.test(residual)

        Jarque Bera Test

data:  residual
X-squared = 2302.4, df = 2, p-value < 2.2e-16
```

P value much smaller than 0.05, so the data is stationary.

There is another function we can use to know how many differentiations need to be stationary

Syntax – ndiffs

ndiffs(Exchangepricetime)

```
> ndiffs(Exchangepricetime)
[1] 1
```

Syntax – nsdiffs   (For seasonality series)

nsdiffs(Exchangepricetime)

```
> nsdiffs(Exchangepricetime)
Error in nsdiffs(Exchangepricetime) : Non seasonal data
```

Because our series is non seasonal.

All Syntax and Functions

```
setwd("D:\CLASSES\4TH TRIMESTAR\BUSINESS VERTICAL\TIME SERIES\ASSIGNMENT\US DOLLAR EXCHANGE RATE")

Exchange_Rate<-read.csv("ExRate.csv")

View(Exchange_Rate)

head(Exchange_Rate)

tail(Exchange_Rate)

summary(Exchange_Rate)

str(Exchange_Rate)

rdate<-as.Date(Exchange_Rate$Date)

fix(rdate)

str(rdate)

plot(Exchange_Rate$US.Dollar,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate")

plot.ts(Exchange_Rate$US.Dollar,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate")

library(ggplot2)

library(scales)

ggplot(data=Exchange_Rate,aes(x=rdate,y=US.Dollar))+geom_line(color="red")+scale_x_date(labels=date_format("%d-%m-%y"))+labs(x="Date",y="US Exchange Rate",title="Date Vs Exchange Rate")

library(xts)

Exchangepricetime=xts(Exchange_Rate$US.Dollar,rdate)

str(Exchangepricetime)
```

```
class(Exchangepricetime)

library(forecast)

library(tseries)

acf(Exchangepricetime)

pacf(Exchangepricetime)

adf.test(Exchangepricetime)

model=auto.arima(Exchangepricetime,ic="aic",trace = TRUE)

acf(ts(model$residuals))

pacf(ts(model$residuals))

myexchangerateforecast=forecast(model,level = c(95),h=10)

myexchangerateforecast

accuracy(myexchangerateforecast)

plot(myexchangerateforecast,xlab="Date",ylab="US Exchange
Rate",main="Date Vs Exchange Rate US Prediction ARIMA (0,1,0)")

Box.test(myexchangerateforecast$resid, lag=5, type= "Ljung-Box")

Box.test(myexchangerateforecast$resid, lag=15, type= "Ljung-Box")

Box.test(myexchangerateforecast$resid, lag=25, type= "Ljung-Box")

model1<-arima(Exchangepricetime,order = c(0,1,0))

model2<-arima(Exchangepricetime,order = c(1,1,1))

model3<-arima(Exchangepricetime,order = c(0,1,1))

model4<-arima(Exchangepricetime,order = c(1,1,0))

summary(model1)
```

model1

residual<-residuals(model1)

gghistogram(residual)+ggtitle("Histogram of Residuals")

qqnorm(residual)

qqline(residual)

jarque.bera.test(residual)

ndiffs(Exchangepricetime)

nsdiffs(Exchangepricetime)

adf.test(myexchangerateforecast$resid)

```
setwd("D:\CLASSES\4TH TRIMESTAR\BUSINESS VERTICAL\TIME SERIES\ASSIGNMENT\US DOLLAR EXCHANGE RATE")
Exchange_Rate<-read.csv("ExRate.csv")
View(Exchange_Rate)
head(Exchange_Rate)
tail(Exchange_Rate)
summary(Exchange_Rate)
str(Exchange_Rate)
rdate<-as.Date(Exchange_Rate$Date)
fix(rdate)
str(rdate)
plot(Exchange_Rate$US.Dollar,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate")
plot.ts(Exchange_Rate$US.Dollar,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate")
library(ggplot2)
library(scales)
ggplot(data=Exchange_Rate,aes(x=rdate,y=US.Dollar))+geom_line(color="red")+scale_x_date(labels=date_format("%d-%m-%y"))
+labs(x="Date",y="US Exchange Rate",title="Date Vs Exchange Rate")
library(xts)
Exchangepricetime=xts(Exchange_Rate$US.Dollar,rdate)
str(Exchangepricetime)
class(Exchangepricetime)
library(forecast)
library(tseries)
acf(Exchangepricetime)
pacf(Exchangepricetime)
adf.test(Exchangepricetime)
model=auto.arima(Exchangepricetime,ic="aic",trace = TRUE)
acf(ts(model$residuals))
pacf(ts(model$residuals))
myexchangerateforecast=forecast(model,level = c(95),h=10)
myexchangerateforecast
accuracy(myexchangerateforecast)
plot(myexchangerateforecast,xlab="Date",ylab="US Exchange Rate",main="Date Vs Exchange Rate US Prediction ARIMA (0,1,0)")
Box.test(myexchangerateforecast$resid, lag=5, type= "Ljung-Box")
Box.test(myexchangerateforecast$resid, lag=15, type= "Ljung-Box")
Box.test(myexchangerateforecast$resid, lag=25, type= "Ljung-Box")
model1<-arima(Exchangepricetime,order = c(0,1,0))
model2<-arima(Exchangepricetime,order = c(1,1,1))
model3<-arima(Exchangepricetime,order = c(0,1,1))
model4<-arima(Exchangepricetime,order = c(1,1,0))
summary(model1)
model1
residual<-residuals(model1)
gghistogram(residual)+ggtitle("Histogram of Residuals")
qqnorm(residual)
qqline(residual)
jarque.bera.test(residual)
ndiffs(Exchangepricetime)
nsdiffs(Exchangepricetime)
adf.test(myexchangerateforecast$resid)
```

Data File – Stock Price, Reliance

(You can download from this Link –

https://finance.yahoo.com/quote/RELIANCE.NS/history/)

1	Date	Open	High	Low	Close	Adj Close	Volume
2	12-08-2021	2124.9	2126.2	2105	2110.5	2110.5	3755507
3	13-08-2021	2117.3	2149.9	2108.95	2145.65	2145.65	5898384
4	16-08-2021	2149.35	2203	2128.15	2173.5	2173.5	10123204
5	17-08-2021	2168.85	2185.2	2147.85	2164.25	2164.25	5841743
6	18-08-2021	2174	2186.8	2152.6	2172.65	2172.65	4650008
7	20-08-2021	2143	2172	2137	2148.25	2148.25	4350228
8	23-08-2021	2174	2174	2132.3	2162.35	2162.35	4547802
9	24-08-2021	2165.05	2192	2155.6	2183.7	2183.7	5475452
10	25-08-2021	2185.4	2220	2180.1	2202.6	2202.6	6175126
11	26-08-2021	2208	2244.9	2205	2230.45	2230.45	8579105
12	27-08-2021	2237	2242.75	2216.05	2227.4	2227.4	4836812
13	30-08-2021	2250	2275.85	2236.8	2270.25	2270.25	6473487
14	31-08-2021	2276.9	2283.75	2242.25	2258.15	2258.15	12223037
15	01-09-2021	2273	2292.9	2263	2267.1	2267.1	5143640
16	02-09-2021	2255	2307.8	2255	2294.4	2294.4	4595048
17	03-09-2021	2310	2395	2302.5	2388.5	2388.5	14151629
18	06-09-2021	2413	2480	2412	2425.6	2425.6	15525644
19	07-09-2021	2430	2458	2412	2440.9	2440.9	8006968
20	08-09-2021	2452	2454	2406.65	2431.35	2431.35	6600210
21	09-09-2021	2427.9	2437.85	2416.1	2425.6	2425.6	4136538
22	13-09-2021	2433	2433	2368.05	2371.55	2371.55	7527598
23	14-09-2021	2375	2394	2366	2368.45	2368.45	4111205
24	15-09-2021	2368.5	2395.75	2368.5	2378.3	2378.3	4186300
25	16-09-2021	2381.55	2436.75	2367	2428.2	2428.2	6206657
26	17-09-2021	2446	2455.85	2375.6	2390.55	2390.55	16098099
27	20-09-2021	2372.1	2418.35	2370	2394.35	2394.35	5436385
28	21-09-2021	2405	2416.6	2384	2404.7	2404.7	4576111

Total No. of Data - 249

(Sample of the Data file and we give the name – stock_price)

We have chosen past data from **12-08-2021** to **11-08-2022**, to see how COVID-19 impact the stock price (Reliance) how we will see the stock price upcoming future, so that we need to do forecasting in R where we will use ARIMA model to see the changes and to predict the stock price.

1st you convert the Excel file to CSV file, so that it is easy to import in R.

Next you should import the file in to R. Before that you should select the path, for path selection you should press CTRL + SHIFT + H, it will be asked the folder where your data file already exist. You should select the folder.

Here you can see that our syntax or path –

Syntax- setwd

```
> setwd("D:/CLASSES/4TH TRIMESTAR/BUSINESS VERTICAL/TIME SERIES/ASSIGNMENT/STOCK PRICE RELIANCE")
```

After that you should declare your data by read.csv

Syntax – read.csv

stock_price<-read.csv("RELIANCE.NS.csv")

If you want to see your data file then you can use view command to see the data.

Syntax - View

View(stock_price)

Stock Price.R | stock_price | Filter

	Date	Open	High	Low	Close	Adj.Close	Volume
1	12-08-2021	2124.90	2126.20	2105.00	2110.50	2110.50	3755507
2	13-08-2021	2117.30	2149.90	2108.95	2145.65	2145.65	5898384
3	16-08-2021	2149.35	2203.00	2128.15	2173.50	2173.50	10123204
4	17-08-2021	2168.85	2185.20	2147.85	2164.25	2164.25	5841743
5	18-08-2021	2174.00	2186.80	2152.60	2172.65	2172.65	4650008
6	20-08-2021	2143.00	2172.00	2137.00	2148.25	2148.25	4350228
7	23-08-2021	2174.00	2174.00	2132.30	2162.35	2162.35	4547802
8	24-08-2021	2165.05	2192.00	2155.60	2183.70	2183.70	5475452
9	25-08-2021	2185.40	2220.00	2180.10	2202.60	2202.60	6175126
10	26-08-2021	2208.00	2244.90	2205.00	2230.45	2230.45	8579105
11	27-08-2021	2237.00	2242.75	2216.05	2227.40	2227.40	4836812
12	30-08-2021	2250.00	2275.85	2236.80	2270.25	2270.25	6473487
13	31-08-2021	2276.90	2283.75	2242.25	2258.15	2258.15	12223037
14	01-09-2021	2273.00	2292.90	2263.00	2267.10	2267.10	5143640

You can use head function for seeing 1st 6 data from your data file.

Syntax - head

head(stock_price)

```
> head(stock_price)
        Date    Open   High     Low   Close Adj.Close   Volume
1 12-08-2021 2124.90 2126.2 2105.00 2110.50   2110.50  3755507
2 13-08-2021 2117.30 2149.9 2108.95 2145.65   2145.65  5898384
3 16-08-2021 2149.35 2203.0 2128.15 2173.50   2173.50 10123204
4 17-08-2021 2168.85 2185.2 2147.85 2164.25   2164.25  5841743
5 18-08-2021 2174.00 2186.8 2152.60 2172.65   2172.65  4650008
6 20-08-2021 2143.00 2172.0 2137.00 2148.25   2148.25  4350228
```

You can use tail function to see last 6 data from your data file.

Syntax – tail

```
> tail(stock_price)
          Date   Open    High     Low   Close Adj.Close  Volume
244 03-08-2022 2600.0 2610.00 2567.45 2606.35   2606.35 6576824
245 04-08-2022 2610.0 2617.75 2535.00 2571.90   2571.90 6676577
246 05-08-2022 2576.0 2578.80 2526.95 2534.00   2534.00 6434433
247 08-08-2022 2531.0 2583.55 2531.00 2567.15   2567.15 4691228
248 10-08-2022 2576.9 2589.90 2557.05 2582.50   2582.50 4949442
249 11-08-2022 2603.1 2609.90 2580.20 2591.10   2591.10 3781795
```

If you want summary of the data then you can use summary function.

Syntax- summary

summary(stock_price)

```
> summary(stock_price)
     Date                Open           High            Low            Close         Adj.Close   
 Length:249         Min.   :2117   Min.   :2126   Min.   :2105   Min.   :2110   Min.   :2110  
 Class :character   1st Qu.:2392   1st Qu.:2416   1st Qu.:2367   1st Qu.:2389   1st Qu.:2389  
 Mode  :character   Median :2471   Median :2500   Median :2445   Median :2475   Median :2475  
                    Mean   :2485   Mean   :2514   Mean   :2456   Mean   :2485   Mean   :2485  
                    3rd Qu.:2587   3rd Qu.:2612   3rd Qu.:2556   3rd Qu.:2585   3rd Qu.:2585  
                    Max.   :2856   Max.   :2856   Max.   :2786   Max.   :2820   Max.   :2820  
     Volume        
 Min.   :  787160  
 1st Qu.: 4836812  
 Median : 6227901  
 Mean   : 6985293  
 3rd Qu.: 8006968  
 Max.   :37841671  
```

Before we proceed further, we should know the structure of our data set. For this we use str function to know the structure.

Syntax – str

str(stock_price)

```
> str(stock_price)
'data.frame':	249 obs. of  7 variables:
 $ Date     : chr  "12-08-2021" "13-08-2021" "16-08-2021" "17-08-2021" ...
 $ Open     : num  2125 2117 2149 2169 2174 ...
 $ High     : num  2126 2150 2203 2185 2187 ...
 $ Low      : num  2105 2109 2128 2148 2153 ...
 $ Close    : num  2110 2146 2174 2164 2173 ...
 $ Adj.Close: num  2110 2146 2174 2164 2173 ...
 $ Volume   : int  3755507 5898384 10123204 5841743 4650008 4350228 4547802 5475452 6175126 8579105 ...
```

summary(stock_price$Close)

```
> summary(stock_price$Close)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   2110    2389    2475    2485    2585    2820 
```

So, for time series we need to convert of date to Date format so that we can proceed further. For this we will use as.Date function and declare a variable rdate.

Syntax – as.Date

rdate<-as.Date(stock_price$Date)

Now we can fix this variable because further it will be not changed.

Syntax- fix

fix(rdate)

Now you can check the structure of the our variable that we declared rdate where the date of the our data set exists.

Syntax – str

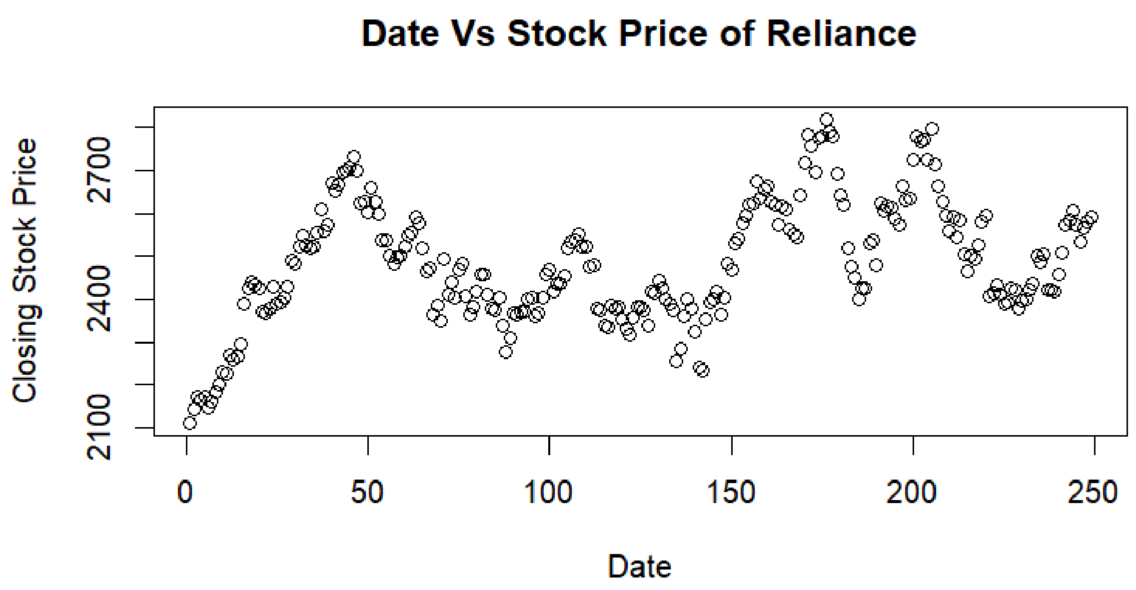
str(rdate)

```
> str(rdate)
 Date[1:249], format: "0012-08-20" "0013-08-20" "0016-08-20" "0017-08-20" "0018-08-20" "0020-08-20" "0023-0
8-20" ...
```

Now you can plot graph **Date Vs Stock Price of Reliance**

Syntax – plot

plot(stock_price$Close,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance")

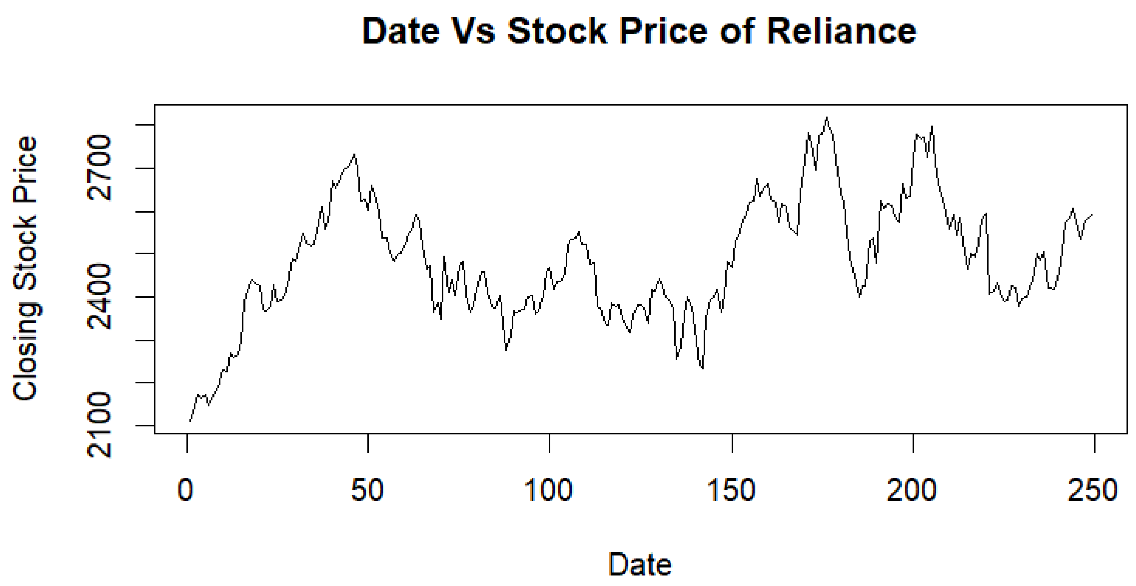


Note - $ sign we use for we want Close from our data set stock_price for plot against Date which is x axis, xlab & ylab used for labelling x axis and y axis "Date" and "Closing Stock Price" respectably, main function we used for what we want to give heading of our graph, here the heading is "Date Vs Stock Price of Reliance".

Now you can plot time series graph for better understanding same Date Vs Stock Price of Reliance which we have drawn before.

Syntax – plot.ts

plot.ts(stock_price$Close,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance")



For better visualization we can draw ggplot which gives us better understanding and better visualization effect. Before that we should import ggplot2 and scales for drawing ggplot.

library(ggplot2)

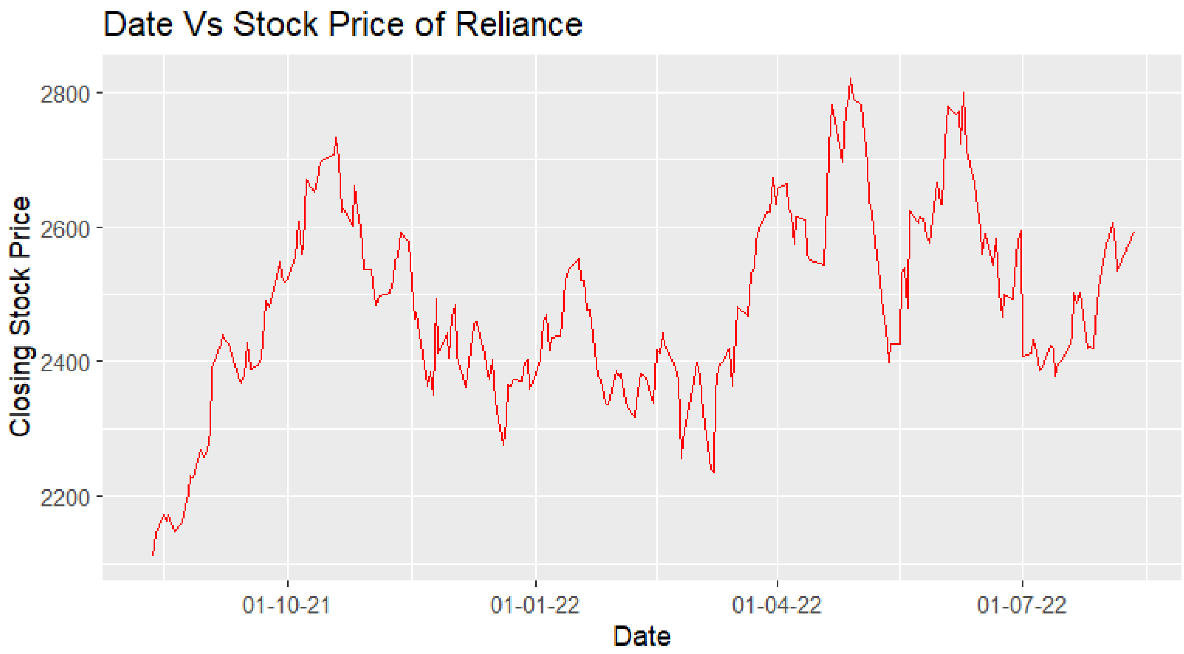
library(scales)

Syntax – ggplot

ggplot(data=stock_price,aes(rdate,Close))+geom_line(color="red")+scale_x_date(labels=date_format("%d-%m-%y"))+labs(x="Date",y="Closing Stock Price",title="Date Vs Stock Price of Reliance")

***Note* – ** data is of our data set, aes denotes what is our x-axes and y-axes to be, geom_line denotes highlighting exactly when changes occur. Here we write in x-axes how we want our date format like (d-m-y) or any format you should mention here and rest labs and title will be same as our previous graphs. Colour also you mention if you want.

Here is our graph –



Now, it's time to convert our data set in to time series format. This step is very important for time series because without this step we can't proceed our prediction which is our goal. For our prediction we need our data set in time series format.

Before that we should import xts function and declare library.

library(xts)

xts function is known Constructor function for creating an extensible time-series object.

Syntax – xts

Stockpricetime=xts(stock_price$Close,rdate)

We gave another name Stockpricetime where we convert both rdate and Close both into time series format.

Now if you want to see the structure of our new data set Stockpricetime then use the function str.

Syntax – str

str(Stockpricetime)

```
> str(Stockpricetime)
An ‘xts’ object on 2021-08-12/2022-08-11 containing:
  Data: num [1:249, 1] 2110 2146 2174 2164 2173 ...
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
 NULL
```

For better understand you can use class function also –

Syntax- class

```
> class(Stockpricetime)
[1] "xts" "zoo"
```

Now you can see our data file is converted into time series format and the name of our new data set is Stockpricetime.

Now come to forecast session which is our objective

Before we forecast, we should import tseries and forecast package from our library.

library(forecast)

library(tseries)

Before we go further we should know either our data is stationary or not for that we can check acf, pacf and adf test.

acf test (Autocorrelation function)-

Syntax – acf

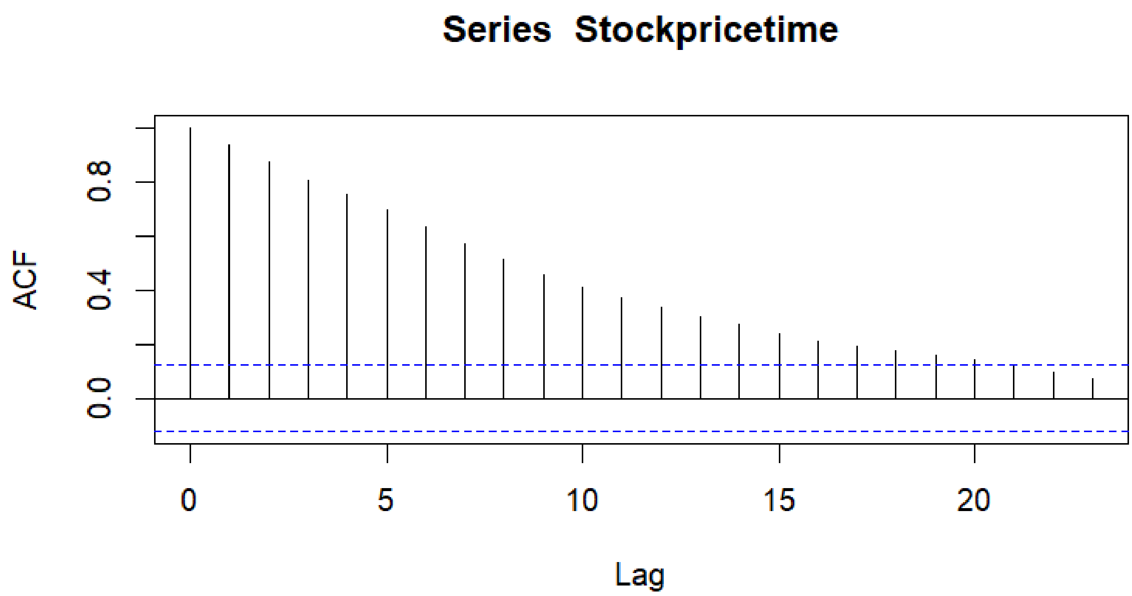
acf(Stockpricetime)

The two blue lines are called significance levels and the interpretation is for stationarity all the lines come should come under these two blue lines but here all the lines are above the blue lines. So, the data is not stationary.

We can also test pacf to know stationary –

Syntax- pacf

pacf = (Partial Autocorrelation Function)



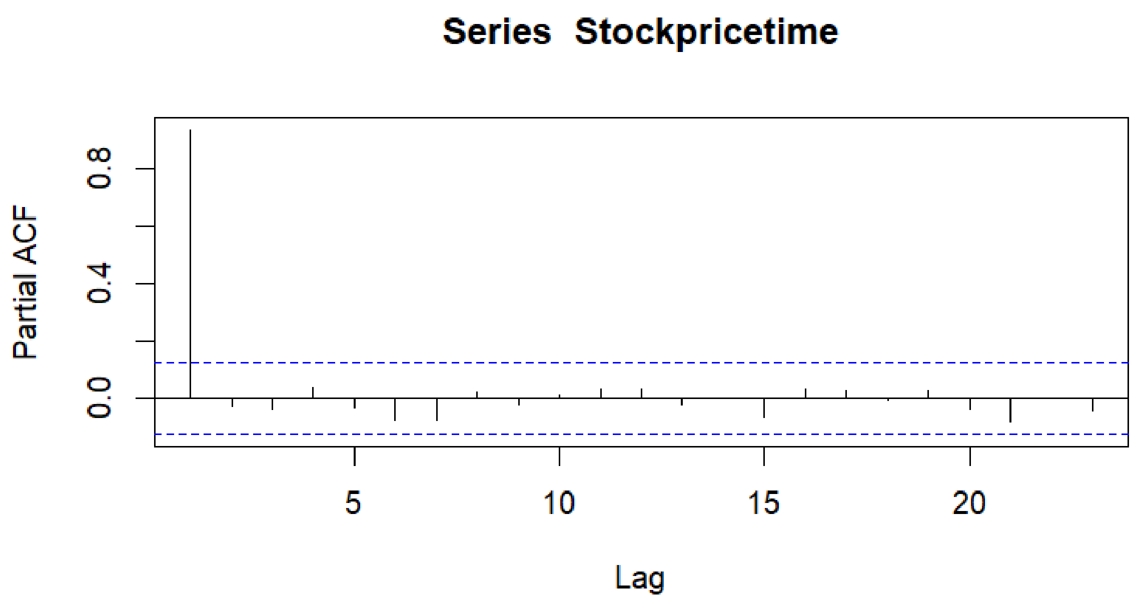
The two blue lines are called significance levels and the interpretation is for stationarity all the lines come should come under these two blue lines but here all the lines are above the blue lines. So, the data is not stationary.

We can also test pacf to know stationary –

Syntax- pacf

pacf = (Partial Autocorrelation Function)

pacf(Stockpricetime)



Here also we drawn the same interpretation.

Now we can test the adf

adf test (Augmented Dickey - Fuller test)

Syntax – adf.test

adf.test(Stockpricetime)

```
> adf.test(Stockpricetime)

        Augmented Dickey-Fuller Test

data:  Stockpricetime
Dickey-Fuller = -3.4617, Lag order = 6, p-value = 0.04698
alternative hypothesis: stationary
```

So all the above test represent that our data is not stationary.

Now we can test our ARIMA model

Syntax- arima

model1<-arima(Stockpricetime,order = c(0,1,0))

```
> model1

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


sigma^2 estimated as 2011:  log likelihood = -1295.06,  aic = 2592.12
```

model2<-arima(Stockpricetime,order=c(1,1,1))

```
> model2

Call:
arima(x = Stockpricetime, order = c(1, 1, 1))

Coefficients:
         ar1      ma1
      0.2627  -0.2476
s.e.  1.3859   1.4209

sigma^2 estimated as 2010:  log likelihood = -1295.03,  aic = 2596.06
```

model3<-arima(Stockpricetime,order=c(0,1,1))

```
> model3

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

Coefficients:
         ma1
      0.0143
s.e.  0.0616

sigma^2 estimated as 2010:  log likelihood = -1295.03,  aic = 2594.07
```

model4<-arima(Stockpricetime,order=c(1,1,0))

```
> model4

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

Coefficients:
         ar1
      0.0152
s.e.  0.0634

sigma^2 estimated as 2010:  log likelihood = -1295.03,  aic = 2594.06
```

OR

also, we check though auto arima function. Which will give us best arima model for our data.

Syntax – auto.arima

model=auto.arima(Stockpricetime,ic="aic",trace = TRUE)

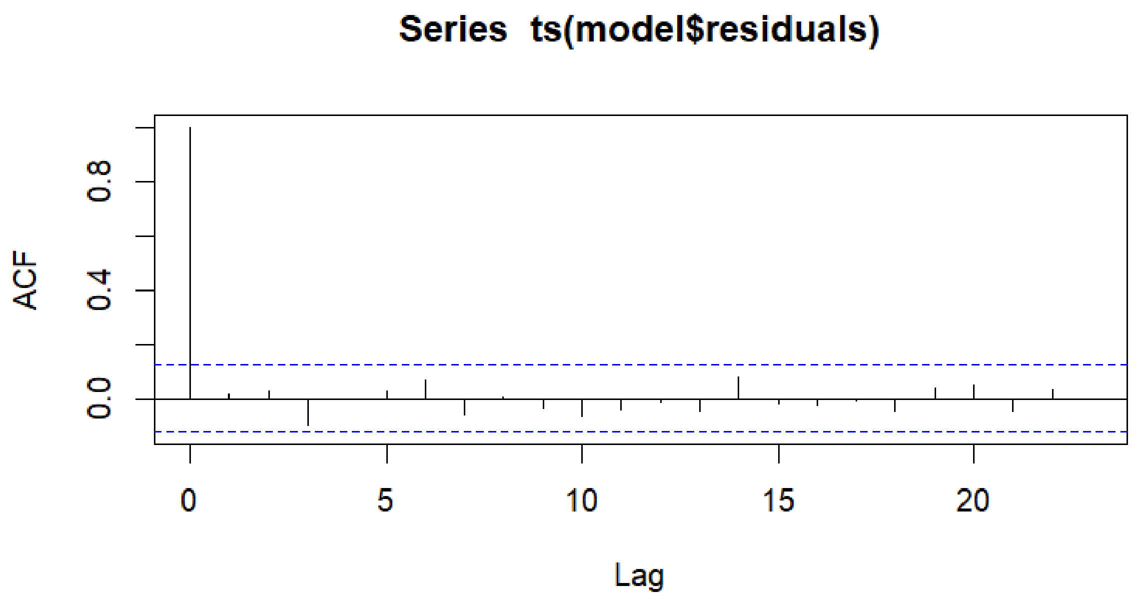
```
> model=auto.arima(Stockpricetime,ic="aic",trace = TRUE)

 Fitting models using approximations to speed things up...

 ARIMA(2,1,2) with drift         : 2593.691
 ARIMA(0,1,0) with drift         : 2586.05
 ARIMA(1,1,0) with drift         : 2588.455
 ARIMA(0,1,1) with drift         : 2588.007
 ARIMA(0,1,0)                    : 2584.514
 ARIMA(1,1,1) with drift         : 2590.298

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

 ARIMA(0,1,0)                    : 2592.12

 Best model: ARIMA(0,1,0)
```

To know best model, we should check the AIC value which is less is known for best model

Now we can also check acf and pacf for stationarity

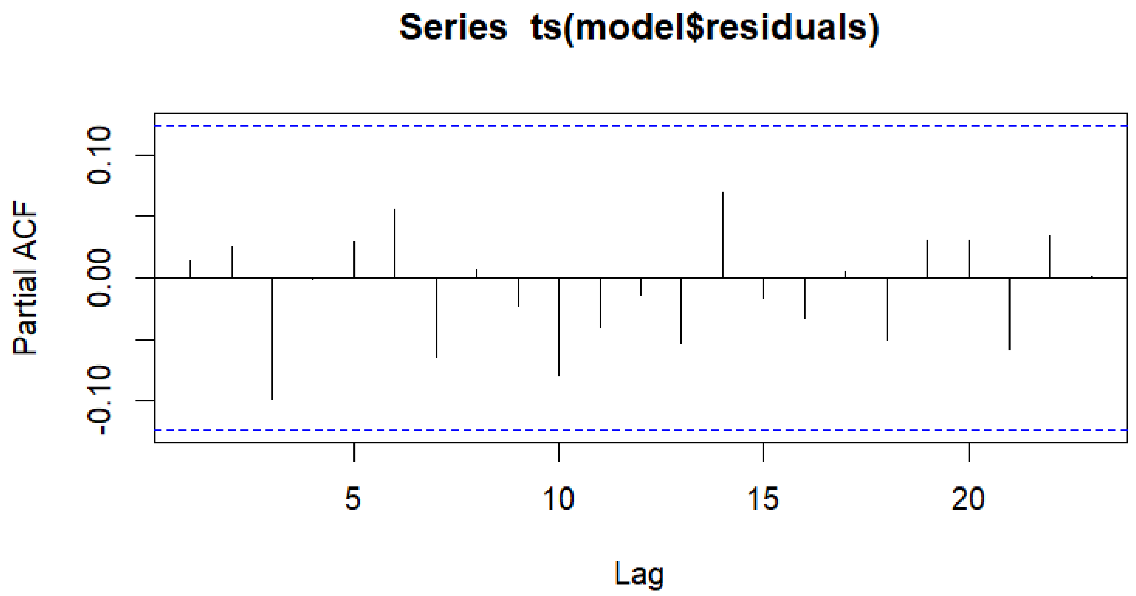
Syntax - acf

acf(ts(model$residuals))



Syntax – pacf

pacf(ts(model$residuals))



Now you can see all the lines with in significance level between blue line. Now you can say that data is stationary.

The final step is our forecasting –

Syntax – forecast

mystockforecast=forecast(model,level = c(95),h=07)

with 95 confidence interval and we want to forecast next 07 days Stock price of Reliance.

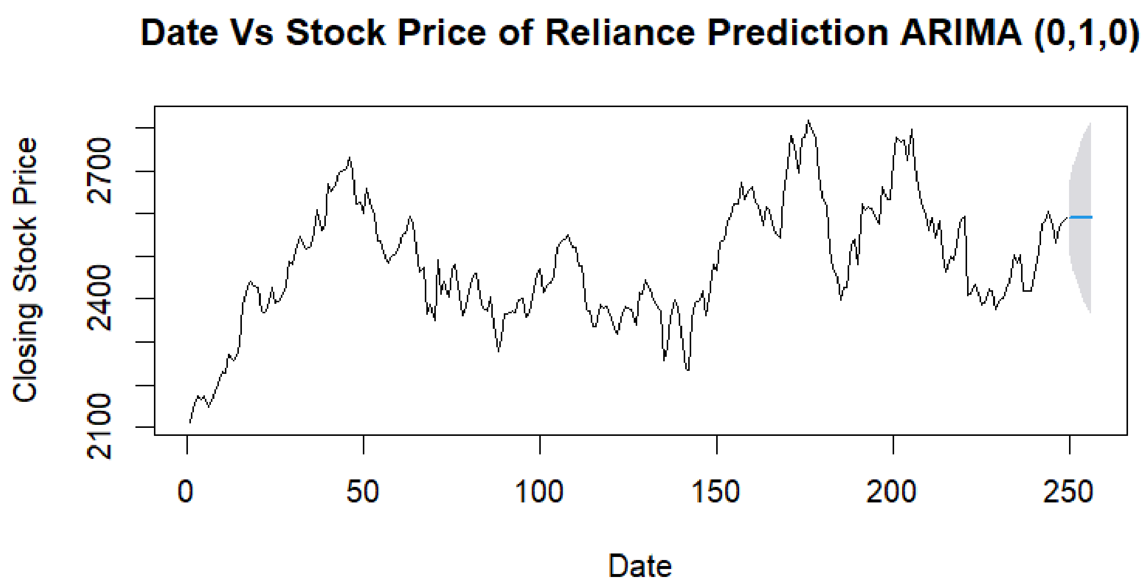
```
> mystockforecast
    Point Forecast    Lo 95    Hi 95
250         2591.1 2503.217 2678.983
251         2591.1 2466.815 2715.386
252         2591.1 2438.882 2743.318
253         2591.1 2415.334 2766.866
254         2591.1 2394.587 2787.613
255         2591.1 2375.831 2806.369
256         2591.1 2358.583 2823.617
```

Now we can forecast our model through graph –

Using plot function

Syntax – plot

plot(mystockforecast,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance Prediction ARIMA (0,1,0)")



Now we check the accuracy of our model –

Syntax – accuracy

accuracy(mystockforecast)

```
> accuracy(mystockforecast)
                   ME     RMSE      MAE        MPE    MAPE      MASE       ACF1
Training set 1.938597 44.74905 33.61049 0.06662606 1.35024 0.9962352 0.01350633
```

Where

ME = Mean Error

RMSE = Root Mean Squared Error

MAE = Mean Absolute Error

MPE = Mean Percentage Error

MAPE = Mean Absolute Percentage Error

MASE = Mean Absolute Scaled Error

Accuracy of our Model = 100 – MAPE = 98.64976

Mean squared error

It is the average square of the difference between the predicted values and actual values. The differences are squared in order to remove the cancellation of positive and negative values with each other.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where,

n= Number of observations

Yi =Actual value of observation

Y(hat)i = Forecasted value of observation

Root Mean Square Deviation

It is the square root value of the Mean squared error.

It gives the output in terms of the metric of the dependent variable. Hence, it is considered more reliable than MSE.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}.$$

Where,

T= Number of observations

y_t =Actual value of observation

y(hat)t = Forecasted value of observation

Mean Absolute Error

It is the average absolute difference between the predicted values and true values. The differences are taken as absolute values in order to remove the cancellation of positive and negative values with each other.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Where,

n= Number of observations

x_i =Actual value of observation

y_i = Forecasted value of observation

Mean Absolute Percentage Error

It is the percentage of the average absolute difference between predicted values and true values, divided by the true value.

Now we want residual for test normality graph – (from our best fit Model)

The power of **Q-Q plots** lies in their ability to summarize any distribution visually.

QQ plots is very useful to determine

- If two populations are of the same distribution
- If residuals follow a normal distribution. Having a normal error term is an assumption in regression and we can verify if it’s met using this.
- Skewness of distribution

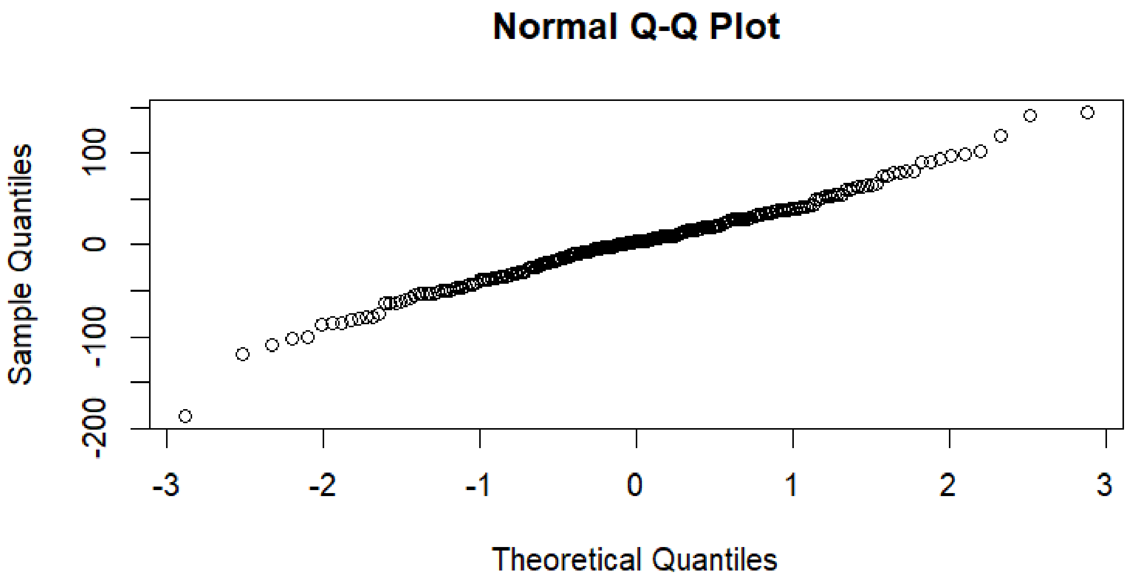
Syntax - residuals

residual<-residuals(model1)

Now we draw the graph

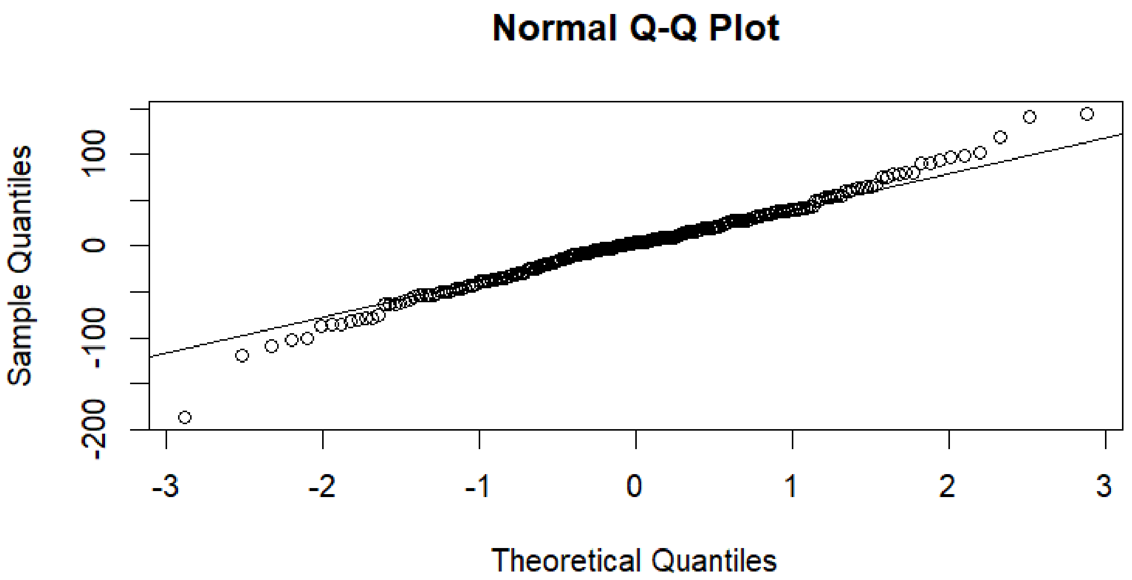
Syntax – qqnorm

qqnorm(residual)



Syntax – qqline

qqline(residual)



The Quantiles-Quantiles plot (Q-Q Plot) is a qualitative way of assessing whether or not sample data could possibly have been drawn from some distribution typically normal distribution.

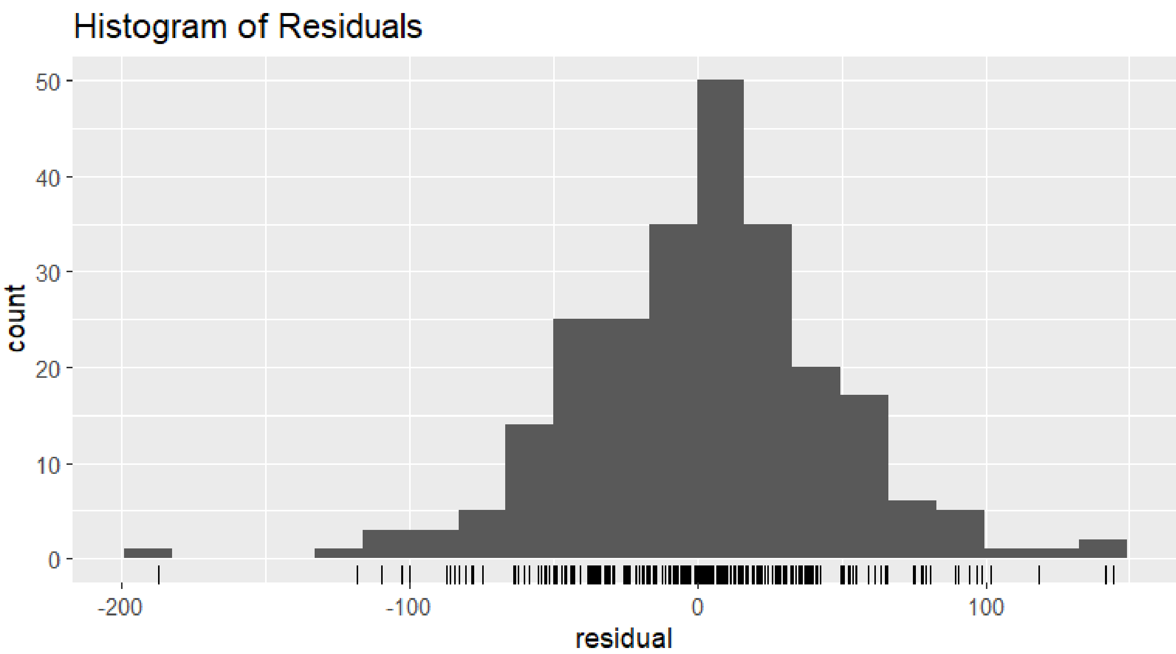
Here our distribution is symmetric with fat tails.

In Q-Q plots, we plot the theoretical Quantile values with the sample Quantile values. Quantiles are obtained by sorting the data. It determines how many values in a distribution are above or below a certain limit.

Also, we can see that histogram of residuals –

Syntax – gghistogram

gghistogram(residual)+ggtitle("Histogram of Residuals")



Here we can say that fat tails of our residuals in normality graph.

Next, we see Ljung-Box Test –

Syntax – Box.test

Box.test(mystockforecast$resid, lag=5, type= "Ljung-Box")

```
> Box.test(mystockforecast$resid, lag=5, type= "Ljung-Box")

        Box-Ljung test

data:  mystockforecast$resid
X-squared = 2.766, df = 5, p-value = 0.736
```

The Ljung-Box test uses the following hypotheses:

H_0: The residuals are independently distributed.

H_A: The residuals are not independently distributed; they exhibit serial correlation.

Ideally, we would like to fail to reject the null hypothesis. That is, we would like to see the p-value of the test be greater than 0.05 because this means the residuals for our time series model are independent, which is often an assumption we make when creating a model.

The test statistic of the test is Q = **2.766** and he p-value of the test is **0.736**, which is much larger than 0.05. Thus, we fail to reject the null hypothesis of the test and conclude that the data values are independent.

Note that we used a lag value of 05 in this example, but you can choose any value that you would like to use for the lag, depending on your particular situation.

Like –

Box.test(mystockforecast$resid, lag=15, type= "Ljung-Box")

```
> Box.test(mystockforecast$resid, lag=15, type= "Ljung-Box")

        Box-Ljung test

data:  mystockforecast$resid
X-squared = 9.082, df = 15, p-value = 0.8732
```

Box.test(mystockforecast$resid, lag=25, type= "Ljung-Box")

```
> Box.test(mystockforecast$resid, lag=25, type= "Ljung-Box")

        Box-Ljung test

data:  mystockforecast$resid
X-squared = 12.935, df = 25, p-value = 0.9773
```

Other tests also we can do for measuring the stationarity like – adf test and jarque.bera.test

Syntax – adf.test

adf.test(mystockforecast$resid)

```
> adf.test(mystockforecast$resid)

        Augmented Dickey-Fuller Test

data:  mystockforecast$resid
Dickey-Fuller = -5.9734, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(mystockforecast$resid) : p-value smaller than printed p-value
```

P value smaller than 0.05 so, the data is stationary.

Similarly –

Syntax - jarque.bera.test

jarque.bera.test(residual)

```
> jarque.bera.test(residual)

        Jarque Bera Test

data:  residual
X-squared = 21.646, df = 2, p-value = 1.994e-05
```

P value much smaller than 0.05, so the data is stationary.

There is another function we can use to know how many differentiations need to be stationary

Syntax – ndiffs

ndiffs(Stockpricetime)

```
> ndiffs(Stockpricetime)
[1] 1
```

Syntax – nsdiffs (For seasonality series)

nsdiffs(Stockpricetime)

```
> nsdiffs(Stockpricetime)
Error in nsdiffs(Stockpricetime) : Non seasonal data
```

Because our series is non seasonal.

All Syntax and Functions

stock_price<-read.csv("RELIANCE.NS.csv")

View(stock_price)

head(stock_price)

tail(stock_price)

str(stock_price)

summary(stock_price)

summary(stock_price$Close)

rdate<-as.Date(stock_price$Date)

fix(rdate)

str(rdate)

plot(stock_price$Close,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance")

plot.ts(stock_price$Close,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance")

library(ggplot2)

library(scales)

ggplot(data=stock_price,aes(rdate,Close))+geom_line(color="red")+scale_x_date(labels=date_format("%d-%m-%y"))+labs(x="Date",y="Closing Stock Price",title="Date Vs Stock Price of Reliance")

```
library(xts)

Stockpricetime=xts(stock_price$Close,rdate)

str(Stockpricetime)

class(Stockpricetime)

library(forecast)

library(tseries)

acf(Stockpricetime)

pacf(Stockpricetime)

adf.test(Stockpricetime)

ndiffs(Stockpricetime)

nsdiffs(Stockpricetime)

model=auto.arima(Stockpricetime,ic="aic",trace = TRUE)

acf(ts(model$residuals))

pacf(ts(model$residuals))

mystockforecast=forecast(model,level = c(95),h=07)

mystockforecast

accuracy(mystockforecast)

plot(mystockforecast,xlab="Date",ylab="Closing Stock Price",main="Date Vs
Stock Price of Reliance Prediction ARIMA (0,1,0)")

Box.test(mystockforecast$resid, lag=5, type= "Ljung-Box")

Box.test(mystockforecast$resid, lag=15, type= "Ljung-Box")

Box.test(mystockforecast$resid, lag=25, type= "Ljung-Box")

model1<-arima(Stockpricetime,order = c(0,1,0))
```

model2<-arima(Stockpricetime,order=c(1,1,1))

model3<-arima(Stockpricetime,order=c(0,1,1))

model4<-arima(Stockpricetime,order=c(1,1,0))

summary(model1)

model1

residual<-residuals(model1)

qqnorm(residual)

qqline(residual)

gghistogram(residual)+ggtitle("Histogram of Residuals")

jarque.bera.test(residual)

adf.test(mystockforecast$resid)

```
stock_price<-read.csv("RELIANCE.NS.csv")
View(stock_price)
head(stock_price)
tail(stock_price)
str(stock_price)
summary(stock_price)
summary(stock_price$Close)
rdate<-as.Date(stock_price$Date)
fix(rdate)
str(rdate)
plot(stock_price$Close,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance")
plot.ts(stock_price$Close,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance")
library(ggplot2)
library(scales)
ggplot(data=stock_price,aes(rdate,Close))+geom_line(color="red")+scale_x_date(labels=date_format("%d-%m-%y"))
+labs(x="Date",y="Closing Stock Price",title="Date Vs Stock Price of Reliance")
```

```
library(xts)
Stockpricetime=xts(stock_price$Close,rdate)
str(Stockpricetime)
class(Stockpricetime)
library(forecast)
library(tseries)
acf(Stockpricetime)
pacf(Stockpricetime)
adf.test(Stockpricetime)
ndiffs(Stockpricetime)
nsdiffs(Stockpricetime)
model=auto.arima(Stockpricetime,ic="aic",trace = TRUE)
acf(ts(model$residuals))
pacf(ts(model$residuals))
mystockforecast=forecast(model,level = c(95),h=07)
mystockforecast
accuracy(mystockforecast)
plot(mystockforecast,xlab="Date",ylab="Closing Stock Price",main="Date Vs Stock Price of Reliance Prediction ARIMA (0,1,0)")
Box.test(mystockforecast$resid, lag=5, type= "Ljung-Box")
Box.test(mystockforecast$resid, lag=15, type= "Ljung-Box")
Box.test(mystockforecast$resid, lag=25, type= "Ljung-Box")
model1<-arima(Stockpricetime,order = c(0,1,0))
model2<-arima(Stockpricetime,order=c(1,1,1))
model3<-arima(Stockpricetime,order=c(0,1,1))
model4<-arima(Stockpricetime,order=c(1,1,0))
summary(model1)
model1
residual<-residuals(model1)
qqnorm(residual)
qqline(residual)
gghistogram(residual)+ggtitle("Histogram of Residuals")
jarque.bera.test(residual)
adf.test(mystockforecast$resid)
```

References

https://dbie.rbi.org.in/DBIE/dbie.rbi?site=home

Reliance Industries Limited (RELIANCE.NS) Stock Historical Prices & Data - Yahoo Finance

https://www.analyticsvidhya.com/blog/2021/11/performing-time-series-analysis-using-arima-model-in-r/

https://www.springml.com/blog/essential-packages-examining-time-series-data-r/

https://www.tutorialspoint.com/r/r_time_series_analysis.htm#:~:text=R%20language%20uses%20many%20functions,using%20the%20ts()%20function.

https://www.geeksforgeeks.org/time-series-analysis-in-r/