Ex.No:1	MACHINE LEARNING TECHNIQUES FOR PREDICTIVE ANALYTICS

AIM:

To implement Predictive Analysis using R to diagnosis whether or not a person has heart disease based on past medical data.

DATASET DESCRIPTION:

The Cleveland data set of heart disease which is used to build a predictive model that classifies that the person will have heart disease based on certain feature variables. This database is a subset of 14variables. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).

ALGORITHM:

Step 1: Import Dataset.
Step 2: Display the structure of the data set.
Step 3: Label the data set.
Step 4: Remove redundant variables from dataset (Data Preparation).
Step 5: Convert data to numeric format.
Step 6: Identify the rows with missing data.
Step 7: Display the structure of the data after removal of missing data.
Step 8: Transform classes for output.
Step 9: Build Model (Data Splicing).
Step 10: Do Classification by KNN (K Nearest Neighbor) Algorithm to make predictions
Step 11: Display Predictions.

PROCEDURE:

STEP 1: IMPORT DATASET.

data <- read.csv("D://heart.csv")

STEP 2: DISPLAY THE STRUCTURE OF THE DATA SET.

str(data)

'data.frame': 303 obs. of 15 variables:
$ sno : int 1 2 3 4 5 6 7 8 9 10 ...
$ age : int 63 37 41 56 57 57 56 44 52 57 ...
$ sex : int 1 1 0 1 0 1 0 1 1 1 ...
$ cp : int 3 2 1 1 0 0 1 1 2 2 ...
$ trestbps: int 145 130 130 120 120 140 140 120 172 150 ...
$ chol : int 233 250 204 236 354 192 294 263 199 168 ...
$ fbs : int 1 0 0 0 0 0 0 0 1 0 ...
$ restecg : int 0 1 0 1 1 1 0 1 1 1 ...
$ thalach : int 150 187 172 178 163 148 153 173 162 174 ...
$ exang : int 0 0 0 0 1 0 0 0 0 0 ...
$ oldpeak : num 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
$ slope : int 0 0 2 2 2 1 1 2 2 2 ...
$ ca : int 0 0 0 0 0 0 0 0 0 0 ...
$ thal : int 1 2 2 2 2 1 2 3 3 2 ...
$ target : int 1 1 1 1 1 1 1 1 1 1 ...

STEP 4: REMOVE REDUNDANT VARIABLES FROM DATASET (DATA PREPARATION).

> data$sno <- NULL

> str(data)

'data.frame': 303 obs. of 14 variables:
$ age : int 63 37 41 56 57 57 56 44 52 57 ...
$ sex : int 1 1 0 1 0 1 0 1 1 1 ...
$ cp : int 3 2 1 1 0 0 1 1 2 2 ...
$ trestbps: int 145 130 130 120 120 140 140 120 172 150 ...
$ chol : int 233 250 204 236 354 192 294 263 199 168 ...
$ fbs : int 1 0 0 0 0 0 0 0 1 0 ...

$ restecg : int 0 1 0 1 1 1 0 1 1 1 ...

$ thalach : int 150 187 172 178 163 148 153 173 162 174 ...

$ exang : int 0 0 0 0 1 0 0 0 0 0 ...

$ oldpeak : num 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...

$ slope : int 0 0 2 2 2 1 1 2 2 2 ...

$ ca : int 0 0 0 0 0 0 0 0 0 0 ...

$ thal : int 1 2 2 2 2 1 2 3 3 2 ...

$ target : int 1 1 1 1 1 1 1 1 1 1 ...

STEP 5: CONVERT DATA TO NUMERIC FORMAT.

data$testno <- as.numeric(data$testno)

STEP 6: IDENTIFY THE ROWS WITH MISSING DATA.

> data<-data[complete.cases(data),]

STEP 7: DISPLAY THE STRUCTURE OF THE DATA AFTER REMOVAL OF MISSING DATA.

> str(data)

'data.frame': 303 obs. of 14 variables:

$ age : int 63 37 41 56 57 57 56 44 52 57 ...

$ sex : int 1 1 0 1 0 1 0 1 1 1 ...

$ cp : int 3 2 1 1 0 0 1 1 2 2 ...

$ trestbps: int 145 130 130 120 120 140 140 120 172 150 ...

$ chol : int 233 250 204 236 354 192 294 263 199 168 ...

$ fbs : int 1 0 0 0 0 0 0 0 1 0 ...

$ restecg : int 0 1 0 1 1 1 0 1 1 1 ...

$ thalach : int 150 187 172 178 163 148 153 173 162 174 ...

$ exang : int 0 0 0 0 1 0 0 0 0 0 ...

$ oldpeak : num 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...

$ slope : int 0 0 2 2 2 1 1 2 2 2 ...

$ ca : int 0 0 0 0 0 0 0 0 0 0 ...

$ thal : int 1 2 2 2 2 1 2 3 3 2 ...

$ target : int 1 1 1 1 1 1 1 1 1 1 ...

STEP 8: TRANSFORM CLASSES FOR OUTPUT.

> data$target <- factor(ifelse(data$target == 1, "positive", "negative"))

> str(data)

'data.frame': 303 obs. of 14 variables:
$ age : int 63 37 41 56 57 57 56 44 52 57 ...
$ sex : int 1 1 0 1 0 1 0 1 1 1 ...
$ cp : int 3 2 1 1 0 0 1 1 2 2 ...
$ trestbps: int 145 130 130 120 120 140 140 120 172 150 ...
$ chol : int 233 250 204 236 354 192 294 263 199 168 ...
$ fbs : int 1 0 0 0 0 0 0 0 1 0 ...
$ restecg : int 0 1 0 1 1 1 0 1 1 1 ...
$ thalach : int 150 187 172 178 163 148 153 173 162 174 ...
$ exang : int 0 0 0 0 1 0 0 0 0 0 ...
$ oldpeak : num 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
$ slope : int 0 0 2 2 2 1 1 2 2 2 ...
$ ca : int 0 0 0 0 0 0 0 0 0 0 ...
$ thal : int 1 2 2 2 2 1 2 3 3 2 ...
$ target : Factor w/ 2 levels "negative","positive": 2 2 2 2 2 2 2 2 2 2 ...

STEP 9: BUILD MODEL (DATA SPLICING).

> trainingSet <- data[1:200,1:13]

> testSet <- data[201:303,1:13]

> trainingOutcomes <- data[1:200,14]

> **testOutcomes <- data[201:303,14]**

> library(class)

STEP 10: DO CLASSIFICATION BY KNN (K NEAREST NEIGHBOR) ALGORITHM TO MAKE PREDICTIONS

> predictions <- knn(train = trainingSet, cl = trainingOutcomes, k= 14, test = testSet)

> predictions

[1] positive positive negative positive positive positive positive positive
[9] positive positive positive positive positive positive positive positive
[17] positive positive positive positive positive positive positive positive
[25] positive positive positive positive positive positive positive positive
[33] positive positive positive positive positive positive positive positive
[41] positive positive positive negative positive positive positive positive
[49] positive positive positive positive positive positive positive positive
[57] positive positive positive positive positive positive positive positive
[65] positive positive positive positive positive positive positive positive

Levels: negative positive

STEP 11: DISPLAY PREDICTIONS.

table(testOutcomes, predictions)

	predictions	
testOutcomes	negative	positive
negative	3	100
positive	0	0

AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT: Thus the Machine Learning Techniques for Predictive Analytics is implemented.

Ex.No:2	PREDICTION OF CREDIT RISK USING LINEAR REGRESSION

AIM:

To implement linear regression technique to predict the customer credit risk using credit card data set.

DATASET DESCRIPTION:

This dataset is part of "An Introduction to Statistical Learning with Applications in R" available at http://www-bcf.usc.edu/~gareth/ISL/index.html

ASSUMPTIONS

The following assumptions about the dataset have been made:

- Credit card Balance refers to the average monthly balance across all of the cards owned by a cardholder. This assumption was made as a result of the Cards variable which refers to the number of credit cards owned by a person and has only one associated Balance figure.
- The Balance is calculated as the highest amount incurred on a credit card in a given month. For example, if a cardholder spends $400, $500, and $600 over the course of three months, and each month pays the balance in full, the average balance will be recorded as $500 (i.e. any preliminary balances before the maximum are not taken into account, neither is the final balance of zero).

ALGORITHM:

Step 1: Import the Dataset.
Step 2: Define the categorical variables.
Step 3: Represent the data in Histograms
Step 4: Finding out the Outliers and remove data which is deviating.
Step 5: Examine Missing data from dataset
Step 6: Replacing the Missing data in the dataset

PROCEDURE:

STEP 1: IMPORT THE DATASET.

credit <- read.csv('C:/Users/User/Desktop/credit.csv')

str(loan_data)

```
'data.frame':    29092 obs. of  8 variables:
 $ loan_status    : int  0 0 0 0 0 0 1 0 1 0 ...
 $ loan_amnt      : int  5000 2400 10000 5000 3000 12000 9000 3000 10000 1000 ...
 $ int_rate       : num  10.6 NA 13.5 NA NA ...
 $ grade          : Factor w/ 7 levels "A","B","C","D",..: 2 3 3 1 5 2 3 2 2 4 ...
 $ emp_length     : int  10 25 13 3 9 11 0 3 3 0 ...
 $ home_ownership: Factor w/ 4 levels "MORTGAGE","OTHER",..: 4 4 4 4 4 3 4 4 4 4 ...
 $ annual_inc     : num  24000 12252 49200 36000 48000 ...
 $ age            : int  33 31 24 39 24 28 22 22 28 22 ...
```

STEP 2: EXPLORING THE DATA SET.

Load the gmodels package

library(gmodels)

Cell Contents

```
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|
Total Observations in Table:  29092
         |         0 |         1 |
         |-----------|-----------|
         |     25865 |      3227 |
         |     0.889 |     0.111 |
         |-----------|-----------|
```

CrossTable(loan_data$grade, loan_data$loan_status, prop.r = TRUE, prop.c = FALSE, prop.t = FALSE, prop.chisq = FALSE)

Cell Contents

|-------------------------|
| N |
N / Row Total

Total Observations in Table: 29092

| loan_data$loan_status

loan_data$grade	0	1	Row Total
A	9084	565	9649
	0.941	0.059	0.332
B	8344	985	9329
	0.894	0.106	0.321
C	4904	844	5748
	0.853	0.147	0.198
D	2651	580	3231
	0.820	0.180	0.111
E	692	176	868
	0.797	0.203	0.030
F	155	56	211
	0.735	0.265	0.007
G	35	21	56
	0.625	0.375	0.002

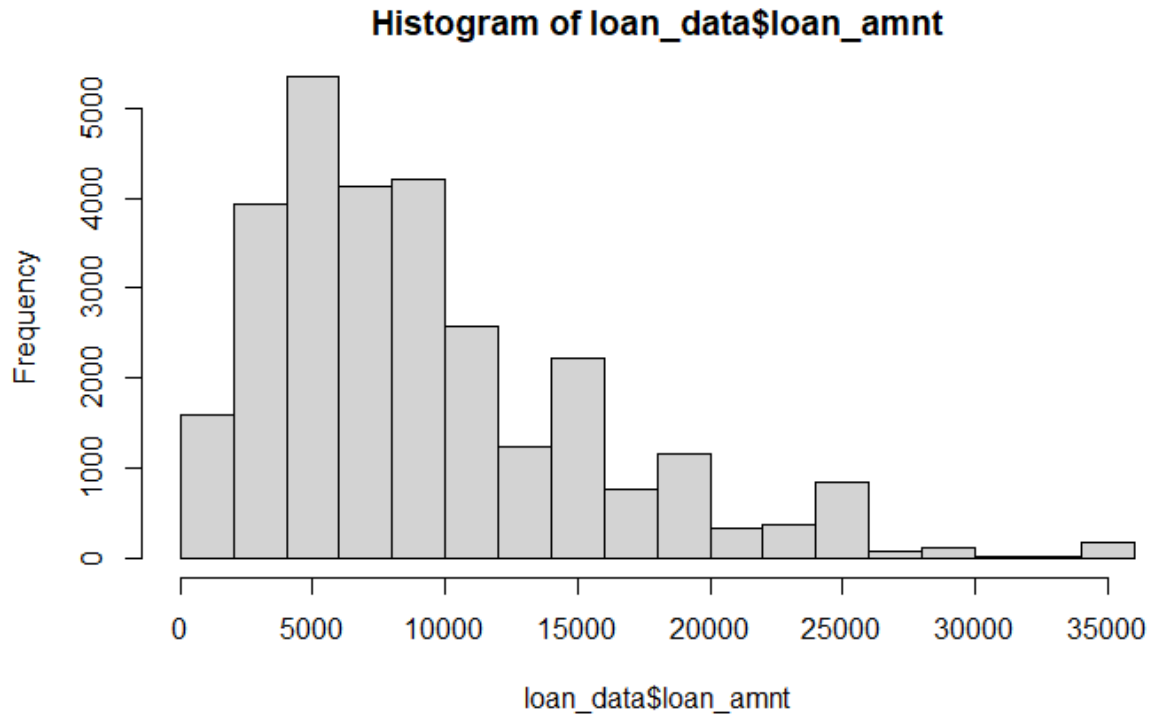
Column Total | 25865 | 3227 | 29092 |

----------|----------|----------|----------|

STEP 3: HISTOGRAMS AND OUTLIERS

Histograms

hist_1 <- hist(loan_data$loan_amnt)

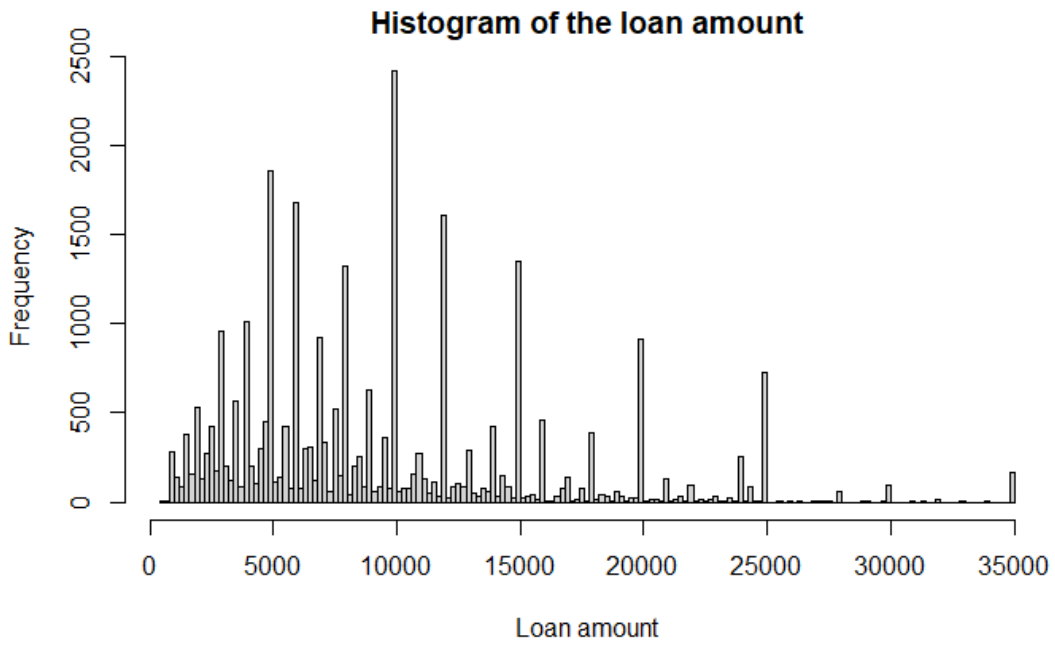


hist_1$breaks

[1] 0 2000 4000 6000 8000 10000 12000 14000 16000 18000 20000 22000 24000 26000 28000 30000

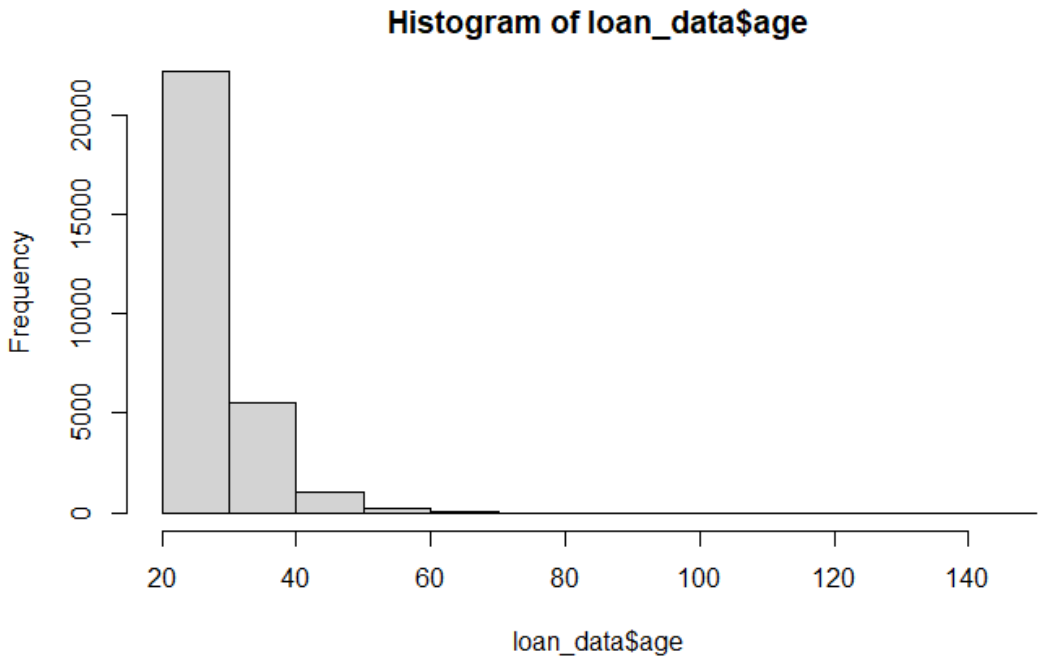
[17] 32000 34000 36000

hist_2 <- hist(loan_data$loan_amnt, breaks = 200, xlab = "Loan amount", main = "Histogram of the loan amount")

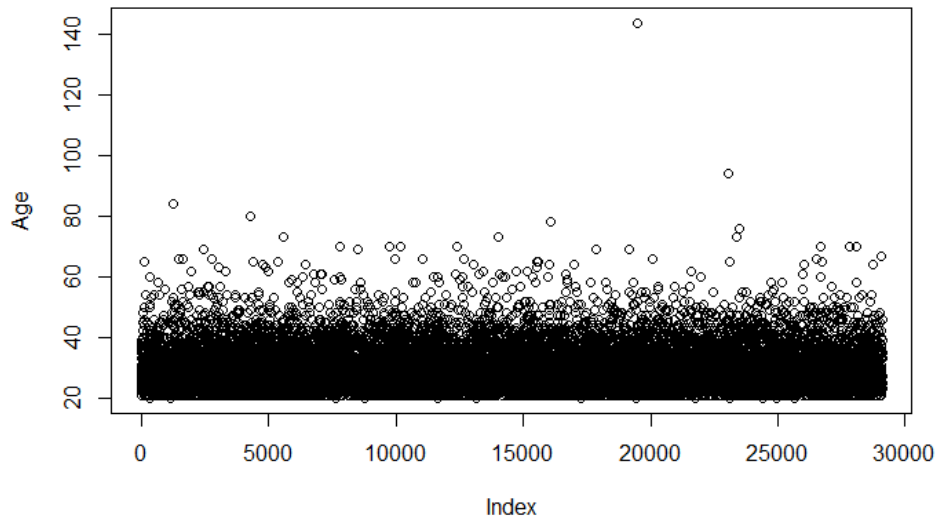


Outliers:

hist(loan_data$age)



plot(loan_data$age, ylab = "Age")



The oldest person in this data set is older than 122 years! Get the index of this outlier using which() and the age of 122 as a cutoff (you can do this using loan_data$age > 122). Assign it to the object index_highage.

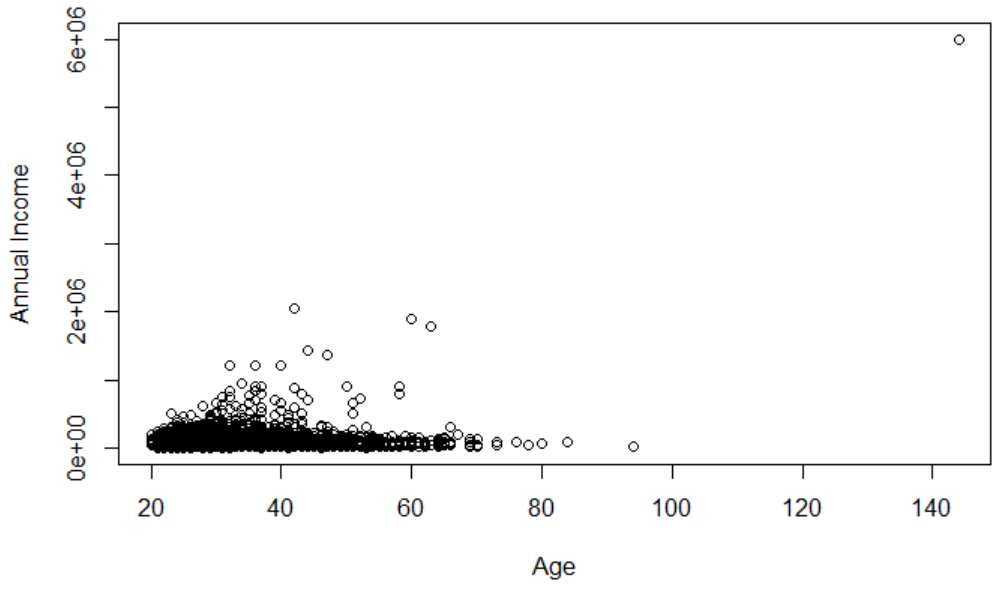
index_highage <- which(loan_data$age > 122)

Create a new data set new_data, after removing the observation with the high age using the object index_highage.

new_data <- loan_data[-index_highage,]

The bivariate scatterplot, with age on the x-axis and annual income on the y-axis.

plot(loan_data$age, loan_data$annual_inc, xlab = "Age", ylab = "Annual Income")



STEP 5: EXAMINE MISSING DATA FROM THE DATASET

The interest rate (int_rate) in the data set loan_data depends on the customer. Unfortunately some observations are missing interest rates. You now need to identify how many interest rates are missing and then delete them.

summary(loan_data$int_rate)

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   5.42    7.90   10.99   11.00   13.47   23.22    2776
```

Get indices of missing interest rates: na_index

na_index <- which(is.na(loan_data$int_rate))

Remove observations with missing interest rates: loan_data_delrow_na

loan_data_delrow_na <- loan_data[-na_index,]

Make copy of loan_data

loan_data_delcol_na <- loan_data

Delete interest rate column from loan_data_delcol_na

loan_data_delcol_na$int_rate <- NULL

Step 5: Replacing Missing data in the dataset

Create an object called median_ir, containing the median of the interest rates in loan_data using the median() function. Include the argument na.rm = TRUE.

Compute the median of int_rate

median_ir <- median(loan_data$int_rate, na.rm = TRUE)

Make copy of loan_data

loan_data_replace <- loan_data

Replace missing interest rates with median

loan_data_replace$int_rate[na_index] <- median_ir

Check if the NAs are gone

summary(loan_data_replace$int_rate)

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.42    8.49   10.99   11.00   13.11   23.22
```

AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT:

Thus the Prediction of Credit Risk Using Linear Regression is implemented in R studio

Ex.No:3	HR ANALYTICS TO PREDICT THE DEMAND FOR HOURLY-EMPLOYEES

AIM:

To apply HR analytics to make a prediction of the demand for hourly-employees for the

following month using R Programming.

ALGORITHM:

Step 1: Import Data using CSV file.

Step 2: Do data cleaning.

Step 3: Renaming the irrelevant variable name.

Step 4: Perform exploratory data analysis.

Step 5: Perform statistical test for correlation.

Step 6: Display distribution plots on the basis of Satisfaction, Evaluation and Average Monthly.

PROCEDURE:

STEP 1: IMPORT DATA USING CSV FILE.

data<-read.csv('D:/hr.csv')

STEP 2: DO DATA CLEANING.

summary(data)

```
 satisfaction_level last_evaluation  number_project  average_montly_hours
 Min.   :0.0900     Min.   :0.3600   Min.   :2.000   Min.   : 96.0
 1st Qu.:0.4400     1st Qu.:0.5600   1st Qu.:3.000   1st Qu.:156.0
 Median :0.6400     Median :0.7200   Median :4.000   Median :200.0
 Mean   :0.6128     Mean   :0.7161   Mean   :3.803   Mean   :201.1
 3rd Qu.:0.8200     3rd Qu.:0.8700   3rd Qu.:5.000   3rd Qu.:245.0
 Max.   :1.0000     Max.   :1.0000   Max.   :7.000   Max.   :310.0
 time_spend_company Work_accident        left        promotion_last_5years
 Min.   : 2.000     Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
 1st Qu.: 3.000     1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
 Median : 3.000     Median :0.0000   Median :0.0000   Median :0.00000
 Mean   : 3.498     Mean   :0.1446   Mean   :0.2381   Mean   :0.02127
 3rd Qu.: 4.000     3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
 Max.   :10.000     Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
    sales              salary
 Length:14999       Length:14999
 Class :character   Class :character
 Mode  :character   Mode  :character
```

STEP 3: RENAMING THE IRRELEVANT VARIABLE NAME.

install.packages("plyr")

library(plyr)

data<-rename(data, c("sales"="role"))

data<-rename(data, c("time_spend_company"="exp_in_company"))

names(data)[10]<-"salary"

head(data)

	satisfaction_level	last_evaluation	number_project	average_montly_hours
1	0.38	0.53	2	157
2	0.80	0.86	5	262
3	0.11	0.88	7	272
4	0.72	0.87	5	223
5	0.37	0.52	2	159
6	0.41	0.50	2	153

	exp_in_company	Work_accident	left	promotion_last_5years	role	salary
1	3	0	1	0	sales	low
2	6	0	1	0	sales	medium
3	4	0	1	0	sales	medium
4	5	0	1	0	sales	low
5	3	0	1	0	sales	low
6	3	0	1	0	sales	low

STEP 4: PERFORM EXPLORATORY DATA ANALYSIS.

dim(data)

[1] 14999 10

str(data)

'data.frame': 14999 obs. of 10 variables:

$ satisfaction_level : num 0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...

$ last_evaluation : num 0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...

$ number_project : int 2 5 7 5 2 2 6 5 5 2 ...

$ average_montly_hours : int 157 262 272 223 159 153 247 259 224 142 ...

$ exp_in_company : int 3 6 4 5 3 3 4 5 5 3 ...

$ Work_accident : int 0 0 0 0 0 0 0 0 0 0 ...

$ left : int 1 1 1 1 1 1 1 1 1 1 ...

$ promotion_last_5years: int 0 0 0 0 0 0 0 0 0 0 ...

$ role : chr "sales" "sales" "sales" "sales" ...

$ salary : chr "low" "medium" "medium" "low" ...

attrition<-as.factor(data$left)

summary(attrition)

0 1

11428 3571

perc_attrition_rate<-sum(data$left/length(data$left))*100

print(perc_attrition_rate)

[1] 23.80825

cor_vars<data[,c("satisfaction_level","last_evaluation","number_project","average_mon tly_hours","exp_in_company","Work_accident","left","promotion_last_5years")]

aggregate(cor_vars[,c("satisfaction_level","last_evaluation","number_project","average_ montly_hours","exp_in_company","Work_accident","promotion_last_5years")], by=list(Category=cor_vars$left), FUN=mean)

```
  Category satisfaction_level last_evaluation number_project
1        0          0.6668096       0.7154734       3.786664
2        1          0.4400980       0.7181126       3.855503
  average_montly_hours exp_in_company Work_accident promotion_last_5years
1              199.0602       3.380032    0.17500875           0.026251313
2              207.4192       3.876505    0.04732568           0.005320638
```

install.packages("reshape2")

library(reshape2)

library(ggplot2)

cor_vars<data[,c("satisfaction_level","last_evaluation","number_project","average_mon tly_hours","exp_in_company","Work_accident","left","promotion_last_5years")]

cor(cor_vars)

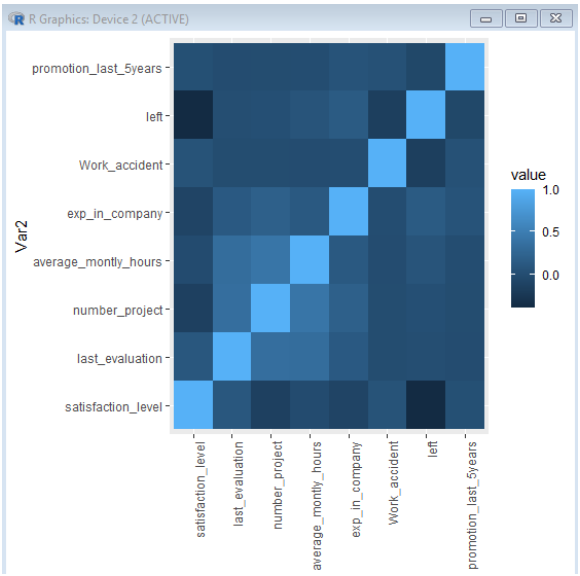
```
                        satisfaction_level last_evaluation number_project
satisfaction_level             1.00000000     0.105021214   -0.142969586
last_evaluation                0.10502121     1.000000000    0.349332589
number_project                -0.14296959     0.349332589    1.000000000
average_montly_hours          -0.02004811     0.339741800    0.417210634
exp_in_company                -0.10086607     0.131590722    0.196785891
Work_accident                  0.05869724    -0.007104289   -0.004740548
left                          -0.38837498     0.006567120    0.023787185
promotion_last_5years          0.02560519    -0.008683768   -0.006063958
                        average_montly_hours exp_in_company Work_accident
satisfaction_level            -0.020048113   -0.100866073    0.058697241
last_evaluation                0.339741800    0.131590722   -0.007104289
number_project                 0.417210634    0.196785891   -0.004740548
average_montly_hours           1.000000000    0.127754910   -0.010142888
exp_in_company                 0.127754910    1.000000000    0.002120418
Work_accident                 -0.010142888    0.002120418    1.000000000
left                           0.071287179    0.144822175   -0.154621634
promotion_last_5years         -0.003544414    0.067432925    0.039245435
                                      left promotion_last_5years
satisfaction_level            -0.38837498           0.025605186
last_evaluation                0.00656712          -0.008683768
number_project                 0.02378719          -0.006063958
average_montly_hours           0.07128718          -0.003544414
exp_in_company                 0.14482217           0.067432925
Work_accident                 -0.15462163           0.039245435
left                           1.00000000          -0.061788107
promotion_last_5years         -0.06178811           1.000000000
```

trans<-cor(cor_vars)

melted_cormat <- melt(trans)

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value))+ geom_tile()

+theme(axis.text.x = element_text(angle = 90, hjust = 1))



STEP 5: PERFORM STATISTICAL TEST FOR CORRELATION.

emp_population_satisfaction <-mean(data$satisfaction_level)

left_pop<-subset(data,left==1)

print(c('The mean for the employee population is: ', emp_population_satisfaction))

[1] "The mean for the employee population is: "

[2] "0.612833522234816"

print(c('The mean for the employees that had a turnover is: ' ,emp_turnover_satisfaction))

[1] "The mean for the employees that had a turnover is: "

[2] "0.440098011761411"

t.test(left_pop$satisfaction_level,mu=emp_population_satisfaction)

```
        One Sample t-test

data:  left_pop$satisfaction_level
t = -39.109, df = 3570, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0.6128335
95 percent confidence interval:
 0.4314385 0.4487576
sample estimates:
mean of x
 0.440098
```

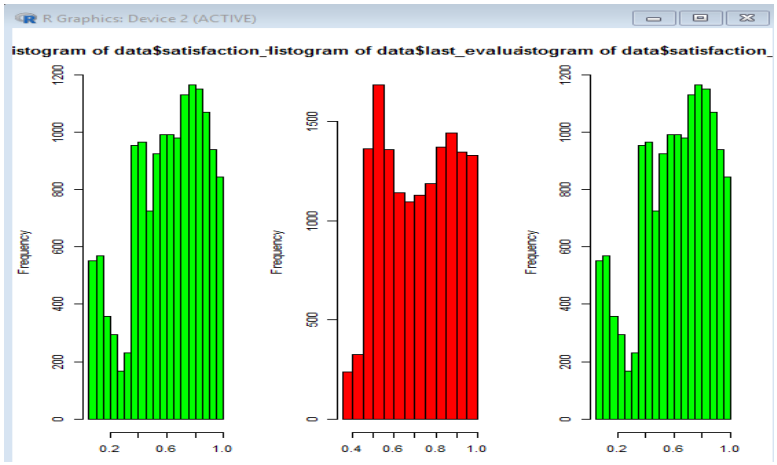
STEP 6: DISPLAY DISTRIBUTION PLOTS ON THE BASIS OF SATISFACTION, EVALUATION AND AVERAGE MONTHLY.

> par(mfrow=c(1,3))

> hist(data$satisfaction_level, col="green")

> hist(data$last_evaluation, col="red")

> hist(data$average_montly_hours, col="blue")



AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT:

Thus the HR Analytics to Predict the Demand for Hourly-Employees is implemented in R Studio

Ex.No:4	APPLY ANALYTICS FOR FORECASTING AIR PASSENGERS

AIM:

The dataset consists of monthly totals of international airline passengers, 1949 to 1960.Main aim is to predict next ten years.

ALGORITHM:

Step 1: Load Data

Step 2: Test the stationarity of time series

Step 3: Make it stationary

Step 4: Time Series Decomposition

Step 5: Model Identification and Estimation

Stepn 6: ARIMA Model Prediction

Step 7: Check normality using Q-Q plot

PROCEDURE:

STEP 1: LOAD DATA

loading library dplyr

library(tseries)

library(forecast)

data(AirPassengers)

AirPassengers

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
```

#The dataset shows the number of passengers travelling on a flight for all the months in a year.

class(AirPassengers)

[1] "ts"

end(AirPassengers)

[1] 1960 12

Check for missing values

sum(is.na(AirPassengers))

[1] 0

cycle of this time series is 12 months in a year

frequency(AirPassengers)

[1] 12

Summary

summary(AirPassengers)

Min. 1st Qu. Median Mean 3rd Qu. Max.

104.0 180.0 265.5 280.3 360.5 622.0

SETP 2: TEST THE STATIONARITY OF TIME SERIES

In order to test the stationarity of the time series, let's run the Augmented Dickey-Fuller Test using the adf.test function from the tseries R package.

First set the hypothesis test:

The null hypothesis : time series is non stationary

The alternative hypothesis : time series is stationary

adf.test(AirPassengers, alternative ="stationary", k=12)

Augmented Dickey-Fuller Test

data: AirPassengers

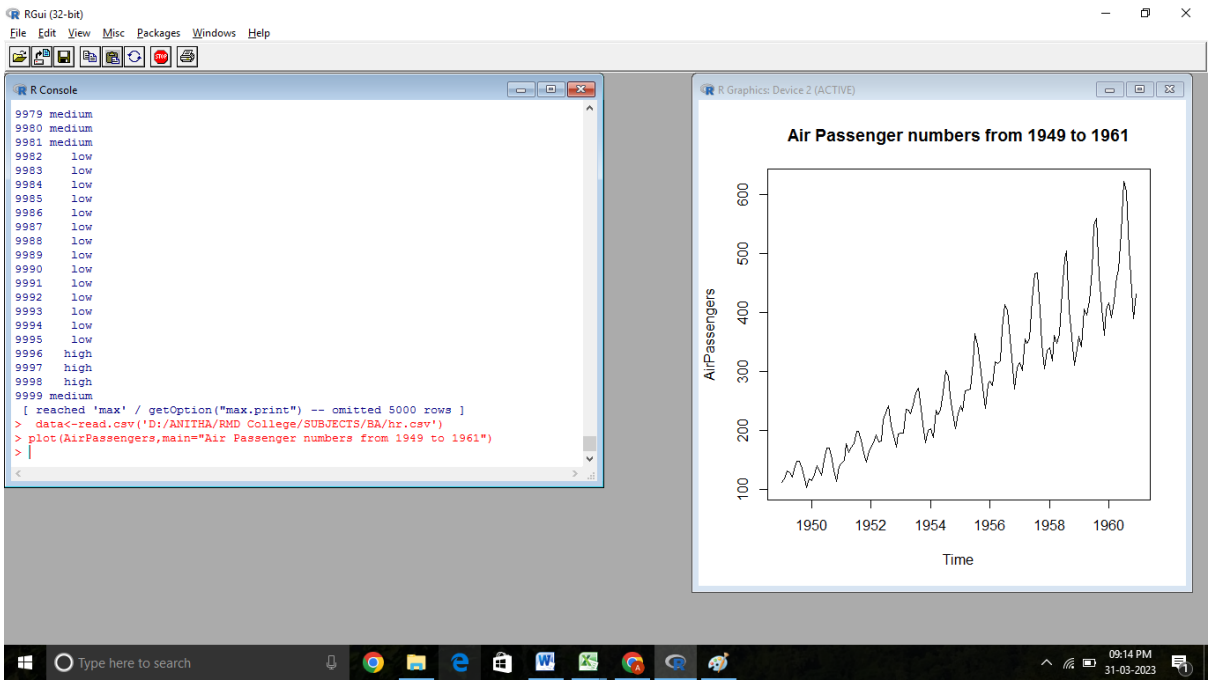
Dickey-Fuller = -1.5094, Lag order = 12, p-value = 0.7807

alternative hypothesis: stationary

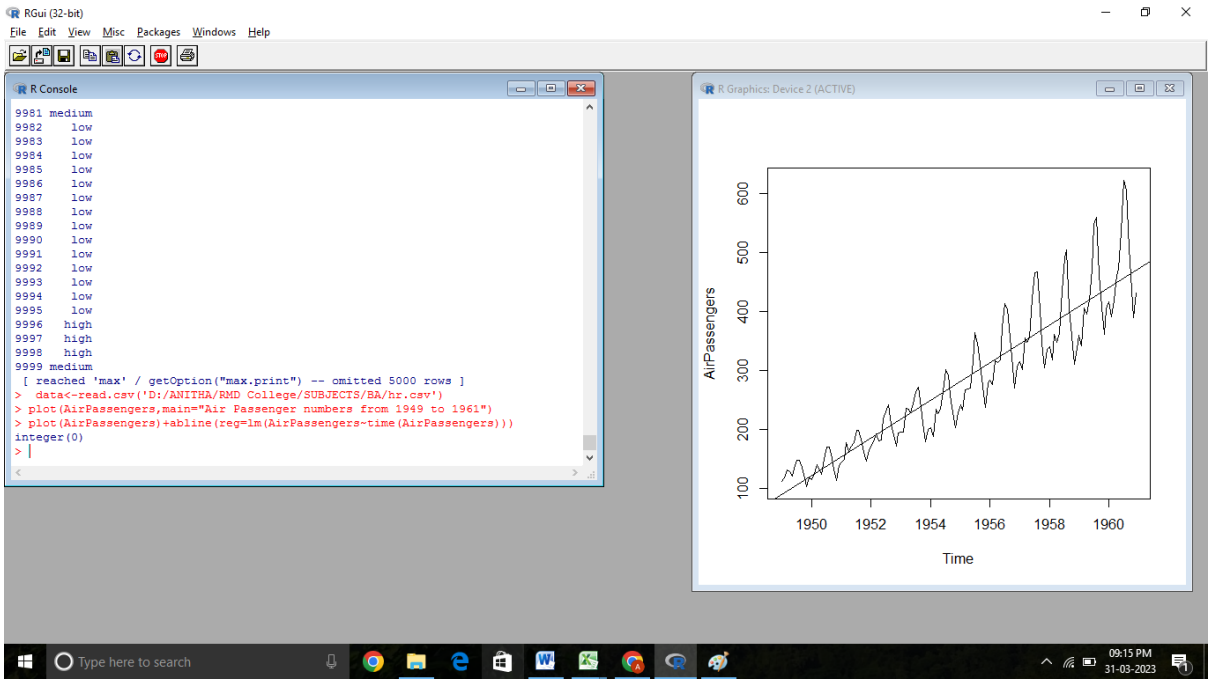
As a rule of thumb, where the p-value is less than 0.05, strong evidence against the null hypothesis, so we reject the null hypothesis. From the above p-value, we concluded that the time series is non-stationary .

#Below plot shows how they vary with time.

plot(AirPassengers,main="Air Passenger numbers from 1949 to 1961") #show sparkine



plot(AirPassengers)+abline(reg=lm(AirPassengers~time(AirPassengers)))



integer(0)

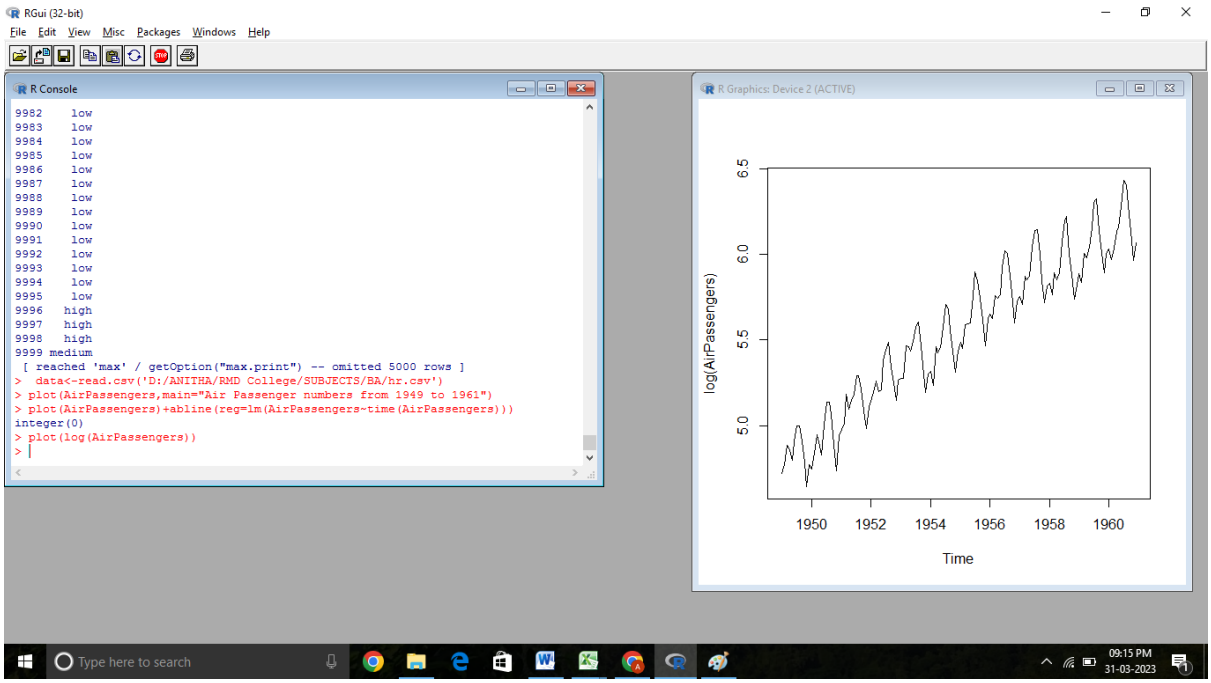
from graph you can see at every point of time variance is changing.Variance means you can see difference between trendline and sparkline.Here variance and mean is also increasing, hence it is not stationary.

STEP 3: MAKE IT STATIONARY

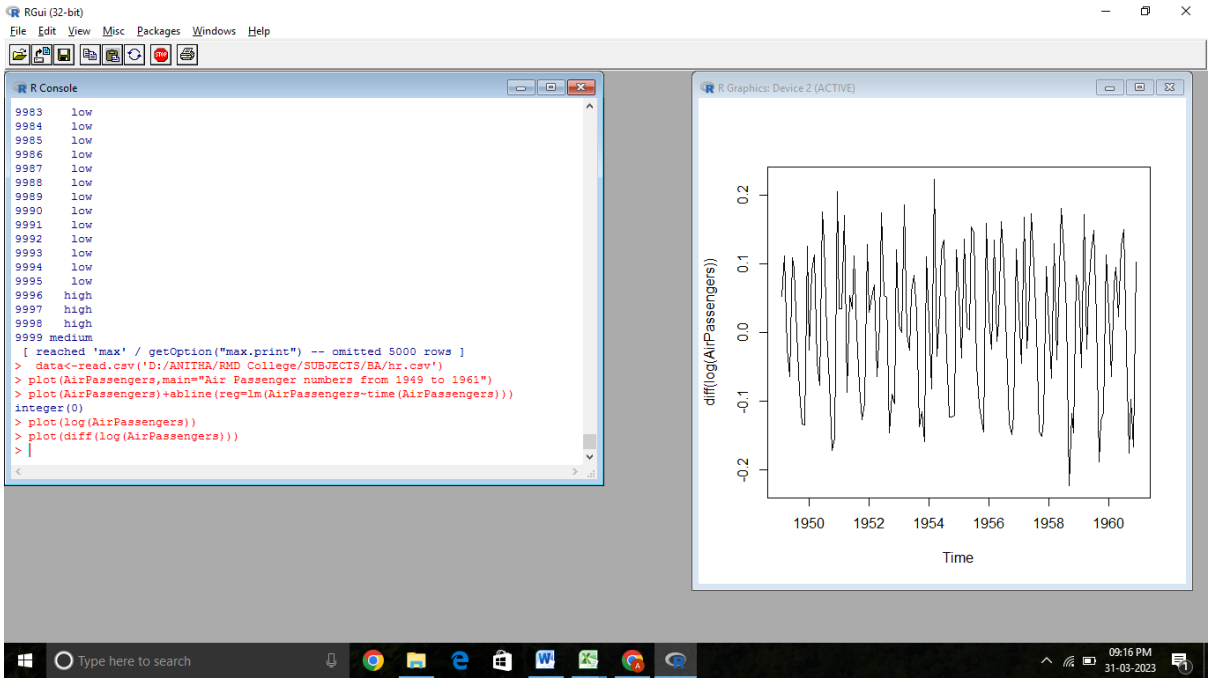
stationary means there should be consitant mean and variance

We need to address two issue before we test stationary series 1.Remove unequal variance using log of time series 2.Address the trend componant, do this by taking difference

plot(log(AirPassengers))



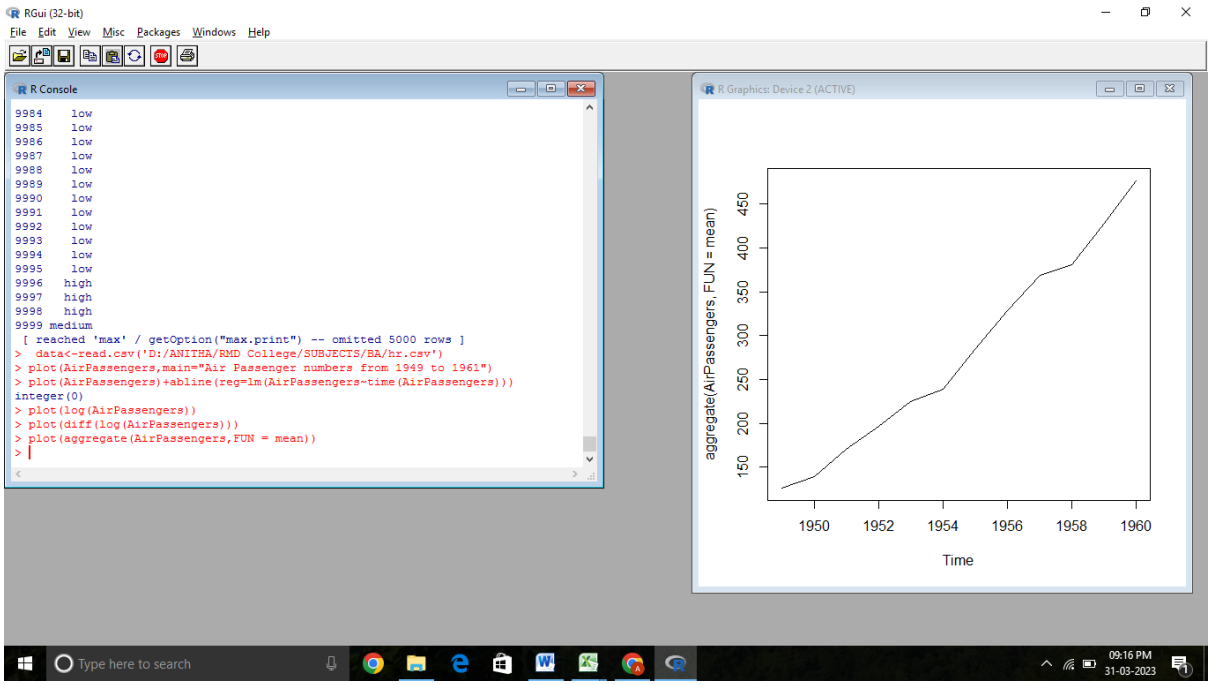
plot(diff(log(AirPassengers)))



#Now you can see mean and variance both are stationary

#Check general trend.

plot(aggregate(AirPassengers,FUN = mean))

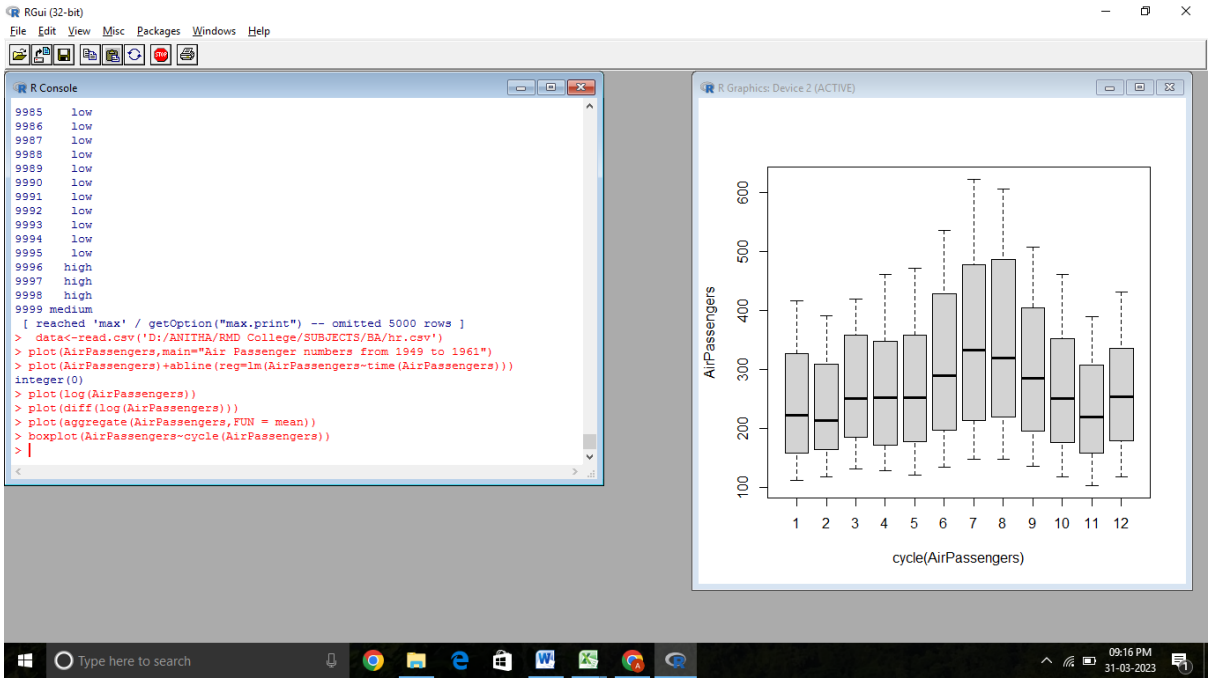


#It is upword trend

Use the boxplot function to see any seasonal effects.

#Show seasonality

boxplot(AirPassengers~cycle(AirPassengers))

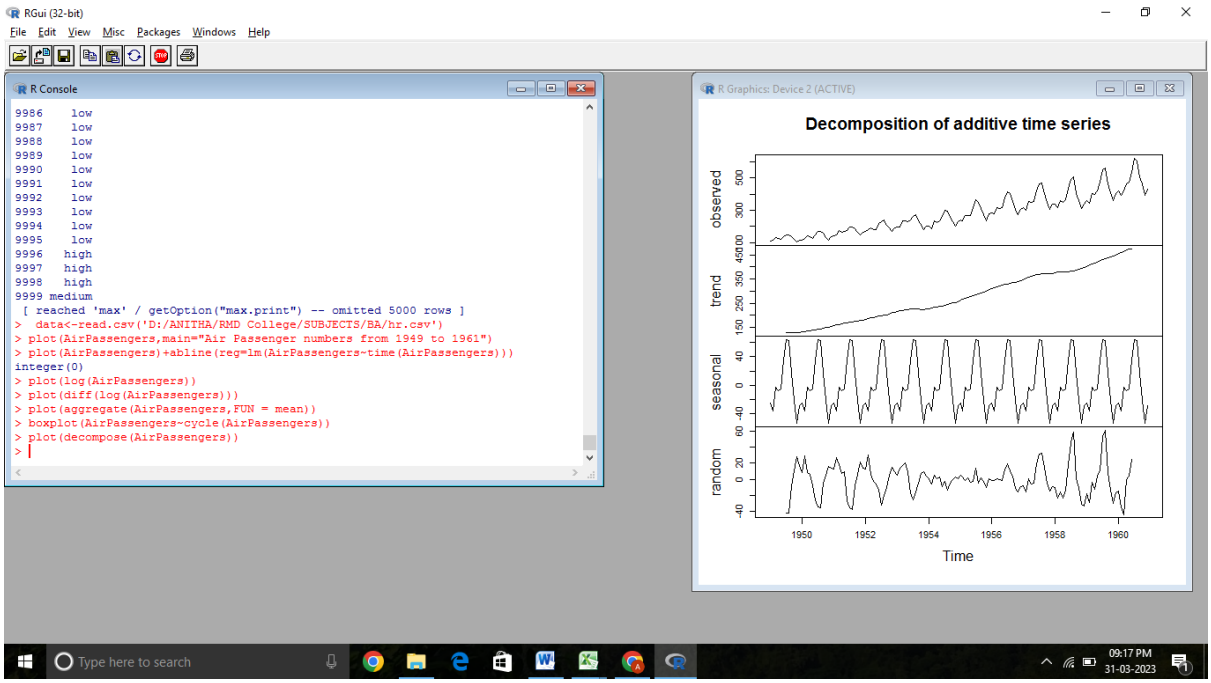


In the boxplot you can see in month july and aug no of passangers traveling are more. The rationale for this could be more people taking holidays and fly over the summer months

STEP 4: TIME SERIES DECOMPOSITION

#Decomposition break data into trend, seasonal, regular and random

plot(decompose(AirPassengers)) # time series decomposition



The above figure shows the time series decomposition into trend, seasonal and random (noise) . It is clear that the time series is non-stationary (has random walks) because of seasonal effects and a trend (linear trend).

STEP 5: MODEL IDENTIFICATION AND ESTIMATION

tsdata <- ts(log(AirPassengers),frequency = 12)

Calculate p d q value

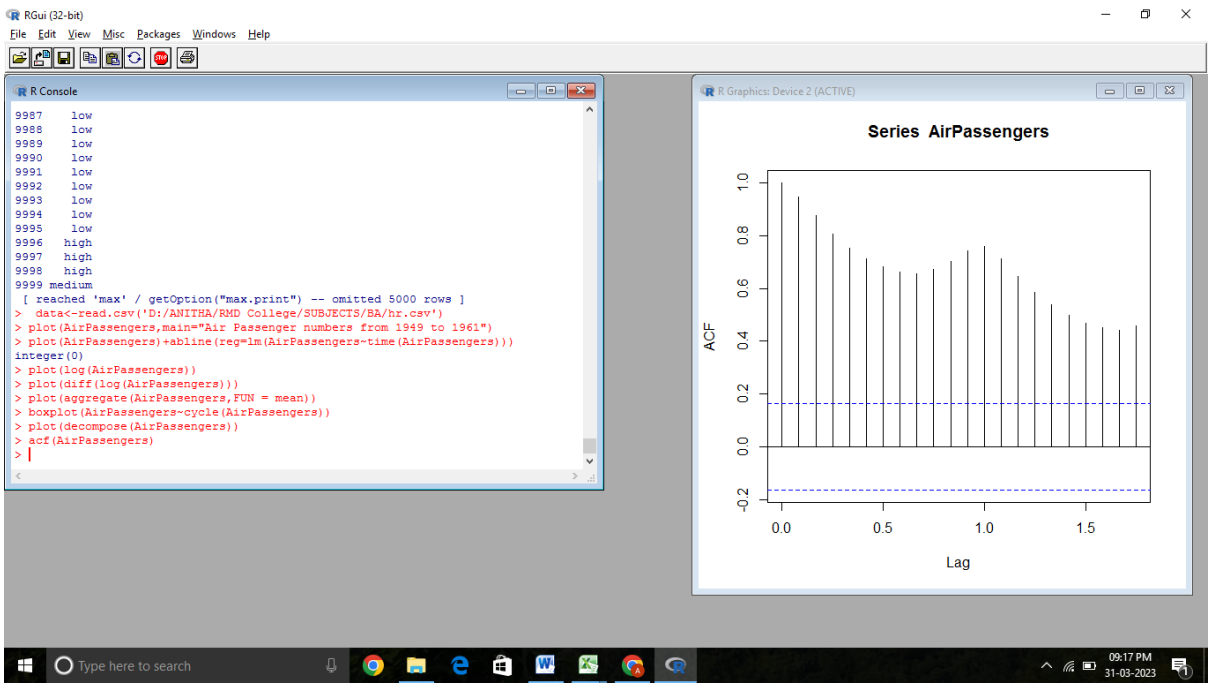
AR(Autoregreesive) :- by seeing the past value, predict own value Integration

MA(Moving Average) :- you take diff intervals and calculate the average Autocorrelation function and partial autocorrelation function to determine value of p and q

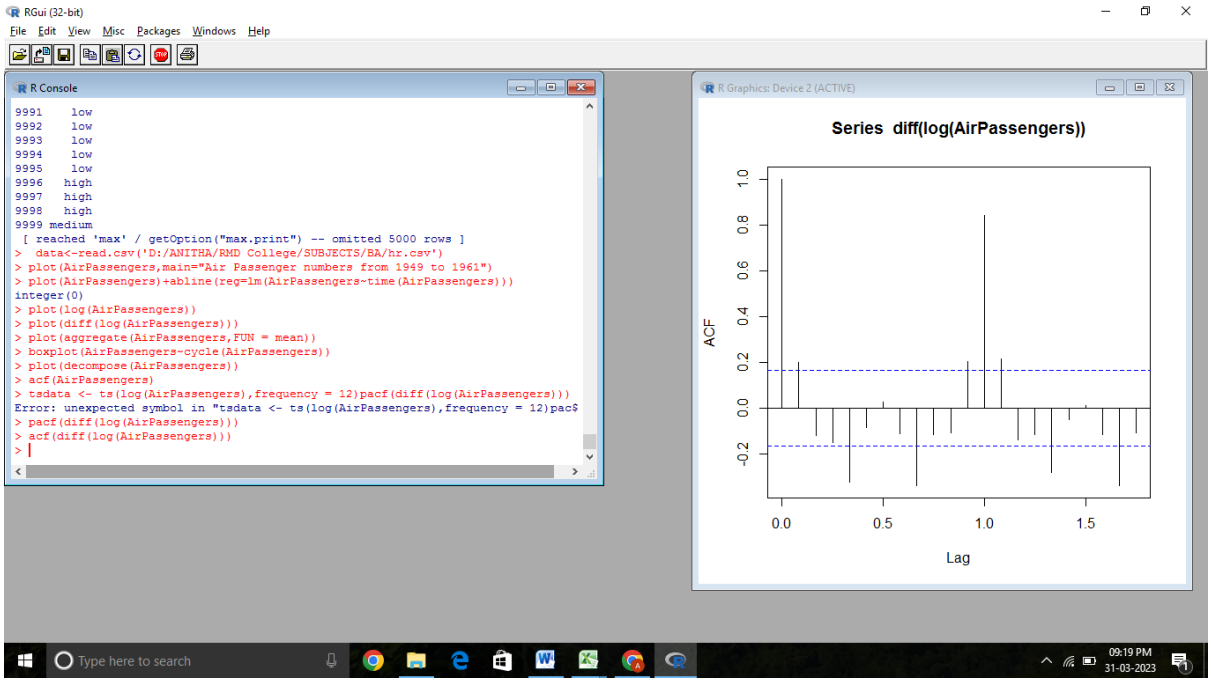
Autocorrelation is the linear dependence of a variable with itself at two points in time Auto correlation function, as the word suggests, auto-correlation, means it is really correlation on itself.With time series we just have a single stream of values, or in other words there is just the X no Y.

So suppose your time series is like this X = 3,5,6,6,7,4,5,6,7,2,3,4,. correlation between 4 and 3, 3 and 2, 2 and 7 (lag 1) will be say y1; correlation between 4 and 2, 3 and 7, 2 and 6 (lag 2) will be say y2; and so on at lags 3 = y3, lag 4 = y4.

acf(AirPassengers)

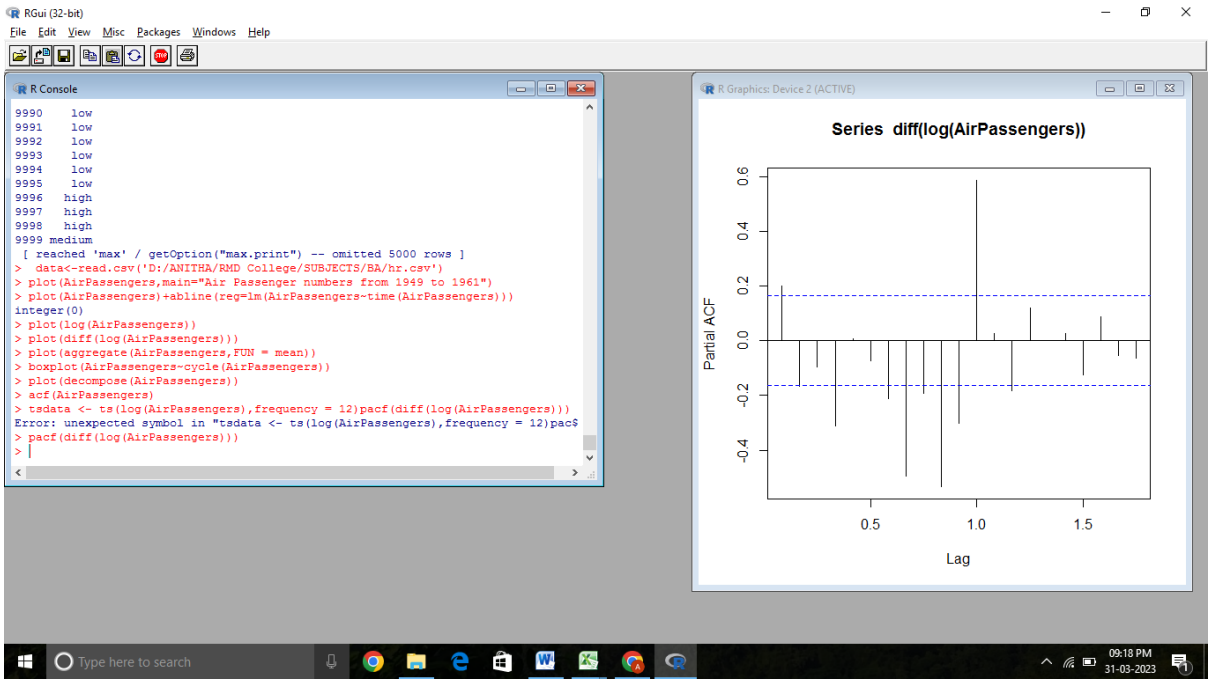


acf(diff(log(AirPassengers)))



Partial auto correlation function, as the word suggests, is partial not complete. Here again we are plot the correlations at various lags 1,2,3 BUT after adjusting for the effects of intermediate numbers.

pacf(diff(log(AirPassengers)))

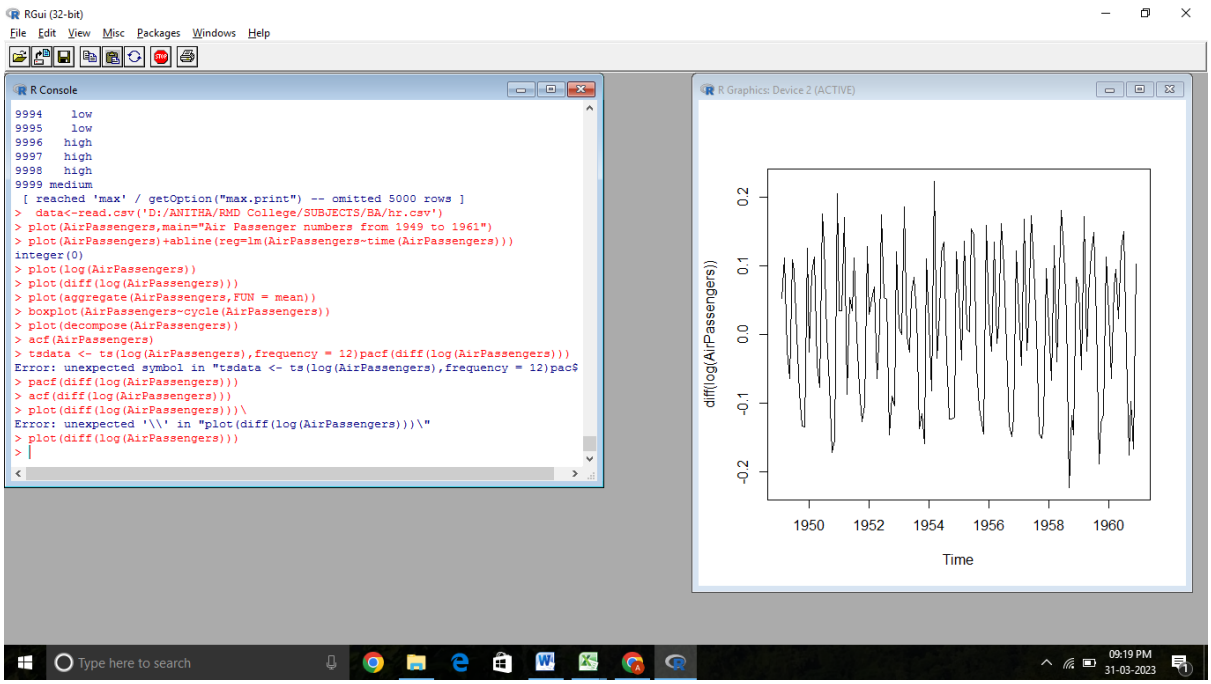


#It determine value of p (value we got as 0)

#d is number of time you do the differentiations to make the mean

#We do diff only one time so value of d is 1

plot(diff(log(AirPassengers)))



STEPN 6: ARIMA MODEL PREDICTION

fit <- arima(log(AirPassengers),c(0,1,1),seasonal = list(order=c(0,1,1),period=12))

fit

Call:

arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0,

1, 1), period = 12))

Coefficients:

ma1 sma1

-0.4018 -0.5569

s.e. 0.0896 0.0731

sigma^2 estimated as 0.001348: log likelihood = 244.7, aic = -483.4

Predict for next 10 years

pred <- predict(fit,n.ahead=10*12) #10 years * 12 months

pred

$pred

Jan Feb Mar Apr May Jun Jul

1961 6.110186 6.053775 6.171715 6.199300 6.232556 6.368779 6.507294
1962 6.206435 6.150025 6.267964 6.295550 6.328805 6.465028 6.603543
1963 6.302684 6.246274 6.364213 6.391799 6.425054 6.561277 6.699792
1964 6.398934 6.342523 6.460463 6.488048 6.521304 6.657526 6.796042
1965 6.495183 6.438772 6.556712 6.584297 6.617553 6.753776 6.892291
1966 6.591432 6.535022 6.652961 6.680547 6.713802 6.850025 6.988540
1967 6.687681 6.631271 6.749210 6.776796 6.810052 6.946274 7.084789
1968 6.783931 6.727520 6.845460 6.873045 6.906301 7.042523 7.181039
1969 6.880180 6.823769 6.941709 6.969295 7.002550 7.138773 7.277288
1970 6.976429 6.920019 7.037958 7.065544 7.098799 7.235022 7.373537

The above output prediction value are in logarithemic part,convert them to original form we need to transform them.

#2.718 is e value and round them to 0 decimal

pred1<-round(2.718^pred$pred,0)

pred1

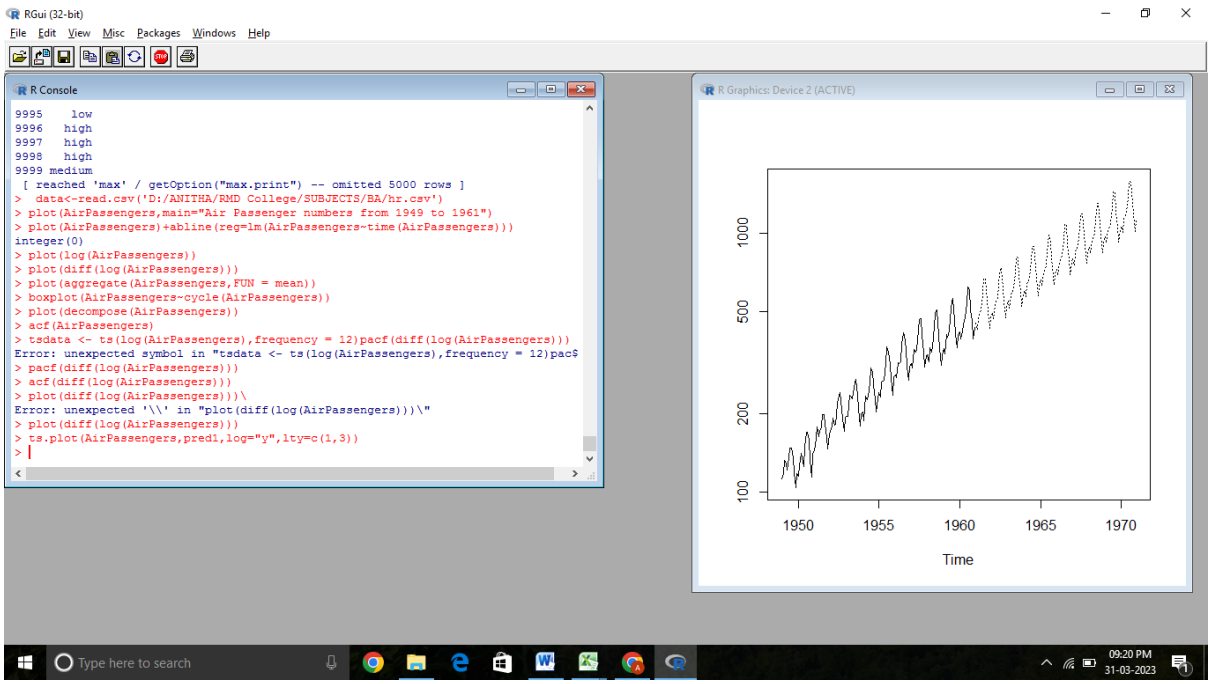
#give op of 1960 to 1970

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1961 450 425 479 492 509 583 670 667 558 497 430 477
1962 496 468 527 542 560 642 737 734 614 547 473 525
1963 546 516 580 597 617 707 812 808 676 602 521 578
1964 601 568 639 657 679 778 894 890 745 663 573 637
1965 661 625 703 723 748 857 984 980 820 730 631 701
1966 728 688 775 796 823 943 1083 1079 903 804 695 772
1967 802 758 853 877 906 1039 1193 1188 994 885 765 850
1968 883 834 939 965 998 1143 1313 1308 1094 975 843 935
1969 972 919 1034 1063 1099 1259 1446 1440 1205 1073 928 1030
1970 1070 1012 1138 1170 1210 1386 1592 1585 1326 1181 1021 1134

plot this model

#line type (lty) can be specified using either text ("blank", "solid", "dashed", "dotted", "dotdash", "longdash", "twodash") or number (0, 1, 2, 3, 4, 5, 6). Note that lty = "solid" is identical to lty=1.

ts.plot(AirPassengers,pred1,log="y",lty=c(1,3))



In above graph, dark(solid) line is original values and dotted are predicted values

Compare predicted values with original values

#Get only 1961 values

data1<-head(pred1,12)

data1

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1961 450 425 479 492 509 583 670 667 558 497 430 477

predicted almost close values to Original values

#Predicted Values

predicted_1960 <- round(data1)#head of Predicted

predicted_1960

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1961 450 425 479 492 509 583 670 667 558 497 430 477

#Original

original_1960 <- tail(AirPassengers,12) #tail of original

original_1960

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Lets Test this Model we are going to take a dataset till 1959, and then we predict value of 1960, then validate that 1960 from alredy existing value we have it in dataset

#Recreate model till 1959

datawide <- ts(AirPassengers, frequency = 12, start=c(1949,1), end=c(1959,12))

datawide

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
```

#Create model

fit1 <- arima(log(datawide),c(0,1,1),seasonal = list(order=c(0,1,1),period=12))

pred <- predict(fit1,n.ahead=10*12) # predictfor now 1960 to 1970

pred1<-2.718^pred$pred

pred1

#give op of 1960 to 1970

```
##          Jan       Feb       Mar       Apr       May       Jun       Jul
```

1960 419.0628 398.6732 466.2820 454.1188 472.9611 546.7614 621.8017

1961 469.5742 446.7270 522.4849 508.8558 529.9692 612.6649 696.7502

1962 526.1740 500.5730 585.4623 570.1903 593.8487 686.5121 780.7325

1963 589.5961 560.9092 656.0306 638.9179 665.4278 769.2603 874.8376

1964 660.6627 628.5180 735.1048 715.9294 745.6347 861.9826 980.2855

1965 740.2952 704.2761 823.7102 802.2235 835.5093 965.8811 1098.4436

1966 829.5262 789.1655 922.9956 898.9190 936.2169 1082.3030 1230.8438

1967 929.5126 884.2870 1034.2482 1007.2696 1049.0631 1212.7576 1379.2027

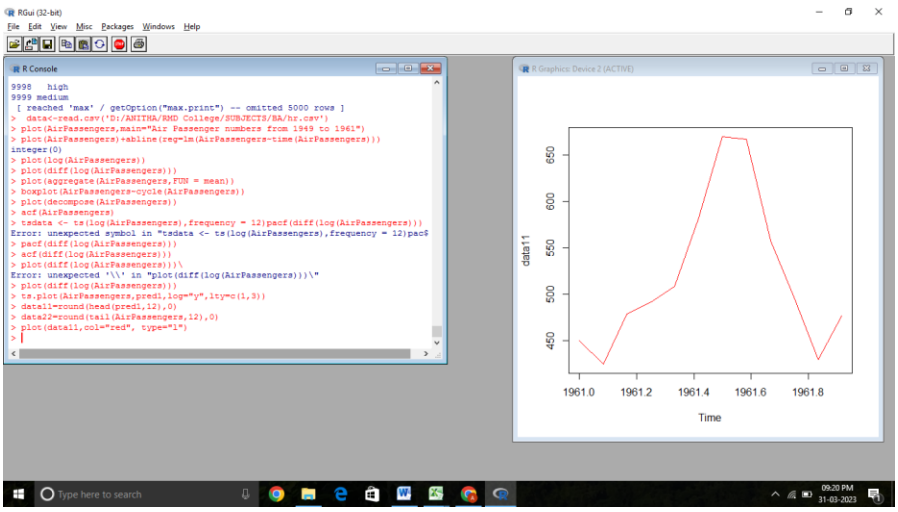
1968 1041.5507 990.8740 1158.9107 1128.6801 1175.5113 1358.9366 1545.4440

1969 1167.0934 1110.3083 1298.5992 1264.7248 1317.2007 1522.7351 1731.7230

data11=round(head(pred1,12),0) #head of Predicted

data22=round(tail(AirPassengers,12),0) #tail of original

plot(data11,col="red", type="l")

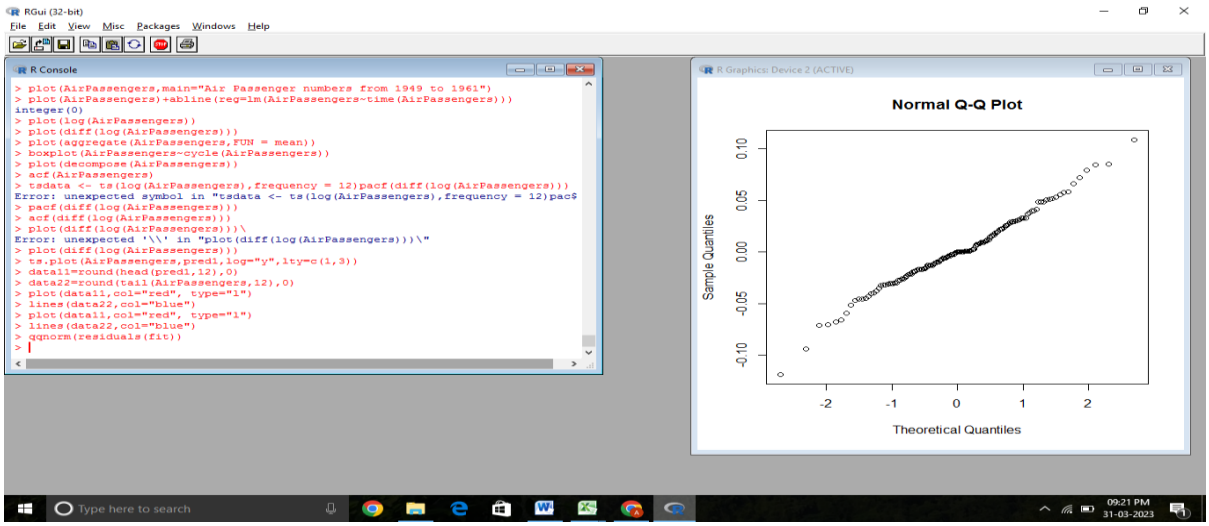


lines(data22,col="blue")

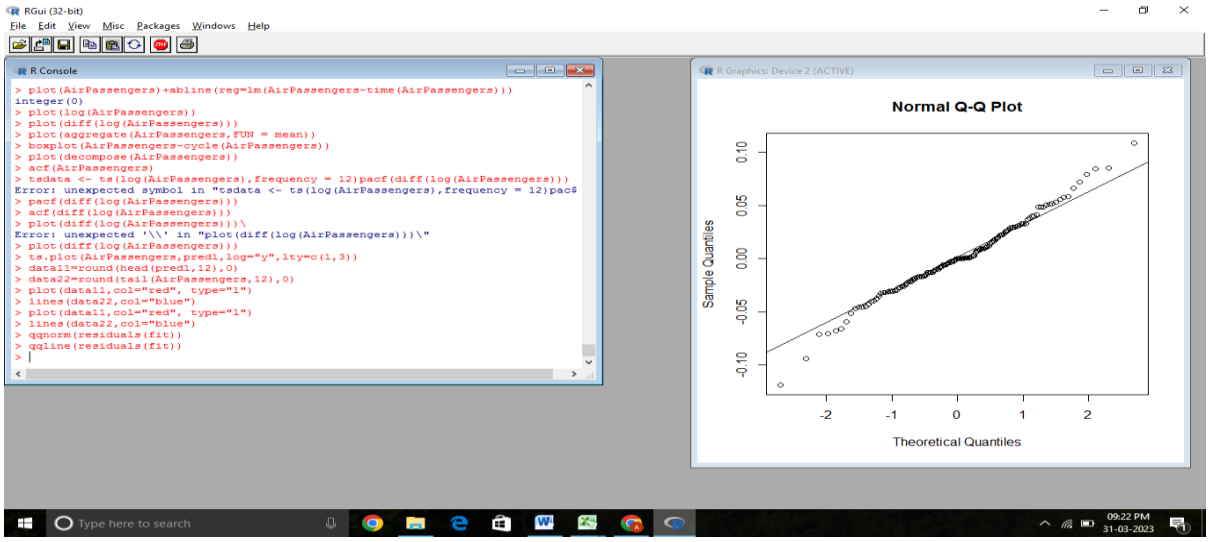
STEP 7: CHECK NORMALITY USING Q-Q PLOT

qqnorm is a generic function the default method of which produces a normal QQ plot of the values in y. qqline adds a line to a "theoretical", by default normal, quantile-quantile plot which passes through the probs quantiles, by default the first and third quartiles.

qqnorm(residuals(fit))



qqline(residuals(fit))



The linearity of the points suggests that the data are normally distributed with mean = 0

AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT:

Thus the Apply Analytics for Forecasting Air Passengers is implemented successfully in R studio.

Ex.No:5	APPLY ANALYTICS FOR FORECASTING AND INVENTORY PLANNING FOR A LARGE RETAILER

AIM:

To apply analytics for forecasting and inventory planning for a large retailer.

DATASET DESCRIPTION

For a long time, Apple such as iPad they look elegant and easy to use. Even the last few product kind disappointed. One thing lead to another, this makes to think how well Apple's stock in the market. With this article, we try to make a model to predict their stock value in the future.

ALGORITHM:

Step 1: Load all necessary libraries

Step 2: Import data and read data from Apple Stock Dataset

Step 3: Check if the data has missing value or not.

Step 4: Cleanse the dataset

PROCEDURE:

STEP 1: LOAD ALL NECESSARY LIBRARIES

```
library(dplyr)      # data wrangling
library(lubridate)  # date manipulation
library(forecast)   # time series library
library(MLmetrics)  # calculate error
library(ggplot2)    # Beautify the graph
library(tidyr)      # Tidy the data
library(zoo)        # Order index observations
library(tseries)    # adf.test
```

STEP 2: IMPORT DATA AND READ DATA FROM APPLE STOCK DATASET

```
# Read data
AAPL <- read.csv("AAPL.csv")
```

 # Check the data

AAPL %>% head()

```
#>         Date     Open     High      Low    Close Adj.Close    Volume
#> 1 1980-12-12 0.128348 0.128906 0.128348 0.128348  0.100178 469033600
#> 2 1980-12-15 0.122210 0.122210 0.121652 0.121652  0.094952 175884800
#> 3 1980-12-16 0.113281 0.113281 0.112723 0.112723  0.087983 105728000
#> 4 1980-12-17 0.115513 0.116071 0.115513 0.115513  0.090160  86441600
#> 5 1980-12-18 0.118862 0.119420 0.118862 0.118862  0.092774  73449600
#> 6 1980-12-19 0.126116 0.126674 0.126116 0.126116  0.098436  48630400
```

change Date datatype. Select the column we need (Date, Close). Then complete the date, so it appears sequentially.

```
AAPL <- AAPL %>%
 mutate(Date = ymd(Date)) %>%
 select(c(Date, Close)) %>%
 complete(Date = seq.Date(min(Date), max(Date), by="day")) %>%
 arrange()
```

AAPL %>% head()

```
#> # A tibble: 6 × 2
#>   Date       Close
#>   <date>     <dbl>
#> 1 1980-12-12 0.128
#> 2 1980-12-13 NA
#> 3 1980-12-14 NA
#> 4 1980-12-15 0.122
#> 5 1980-12-16 0.113
#> 6 1980-12-17 0.116
```

AAPL %>% tail()

#> # A tibble: 6 × 2

#> Date Close

#> <date> <dbl>

#> 1 2022-06-12 NA

#> 2 2022-06-13 132.

#> 3 2022-06-14 133.

#> 4 2022-06-15 135.

STEP 3: CHECK IF THE DATA HAS MISSING VALUE OR NOT.

 # Check missing value

AAPL %>% is.na() %>% colSums()

#> Date Close

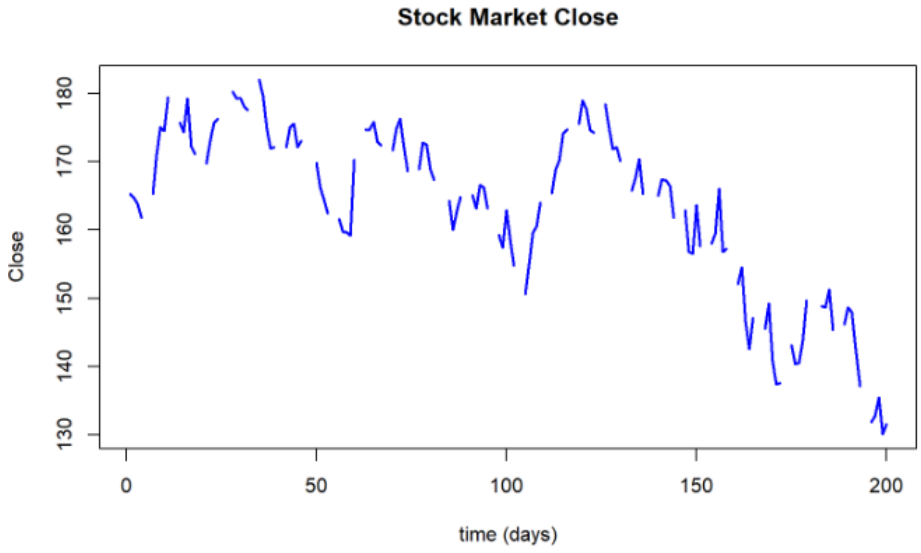
#> 0 4695

The data has missing value

Inspect the data with plot graph

AAPL$Close %>% tail(200) %>%

plot(type="l", col = "blue", lwd = 2, xlab = "time (days)", ylab = 'Close', main = "Stock Market Close")



We cannot use the data like that. Because time series data must appear sequentially. Main requirements for time series data:

(1) The data must be sorted according to the time period from the oldest data to the newest data

(2) The time interval must be the same

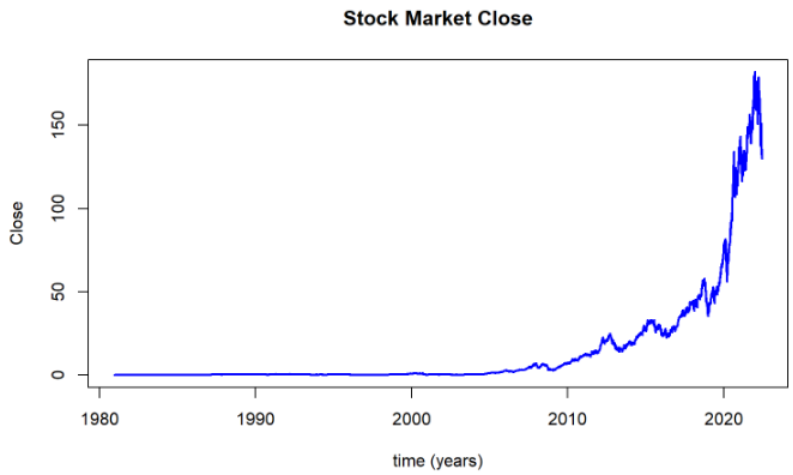
(3) No missing data for each interval

(4) Lastly, nothing can be missing on the dataset

```
library(zoo)

AAPL_new <- AAPL %>%

  mutate(Close = na.fill(Close, "extend")) # Fill the missing value with "extend"

AAPL_new %>%

  tail(200) %>%   # get the last 200 data

  plot(type="l", col = "blue", lwd = 2, xlab = "time (days)", ylab = 'Close', main = "Stock Market Close")
```

Check the whole dataset with plot

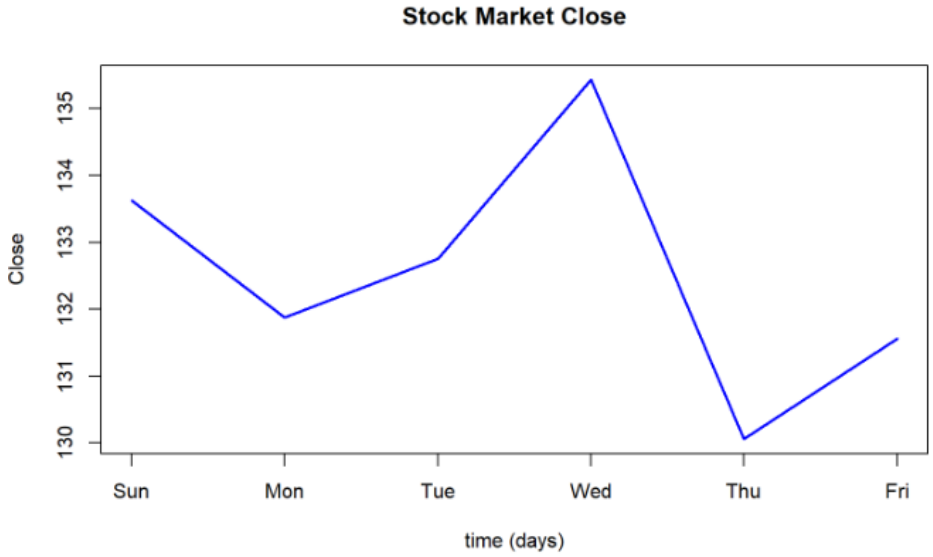
```
AAPL_new %>%

   plot(type="l", col = "blue", lwd = 2, xlab = "time (years)", ylab = 'Close', main = "Stock Market Close")
```



Check the last 6 observations in dataset with plot

AAPL_new %>% tail() %>%

plot(type="l", col = "blue", lwd = 2, xlab = "time (days)", ylab = 'Close', main = "Stock Market Close")



The graph already connected, we can continue the process.

STEP 4: CLEANSE THE DATASET

Check data class AAPL_new

class(AAPL_new)

#> [1] "tbl_df" "tbl" "data.frame"

The dataset has observation with day by day. We could make to month by month. So, the analysis more applicable to the problem.

APPL_m <- AAPL_new %>% group_by(month = lubridate::floor_date(Date, "month")) %>% summarize(Close = mean(Close))

APPL_m %>% head()

```
#> # A tibble: 6 × 2
#>   month      Close
#>   <date>     <dbl>
#> 1 1980-12-01 0.137
#> 2 1981-01-01 0.142
#> 3 1981-02-01 0.118
#> 4 1981-03-01 0.111
#> 5 1981-04-01 0.121
#> 6 1981-05-01 0.131
```

STEP 5: TIME SERIES MODELLING ANALYSIS

Make our data to Object TS / Time Series

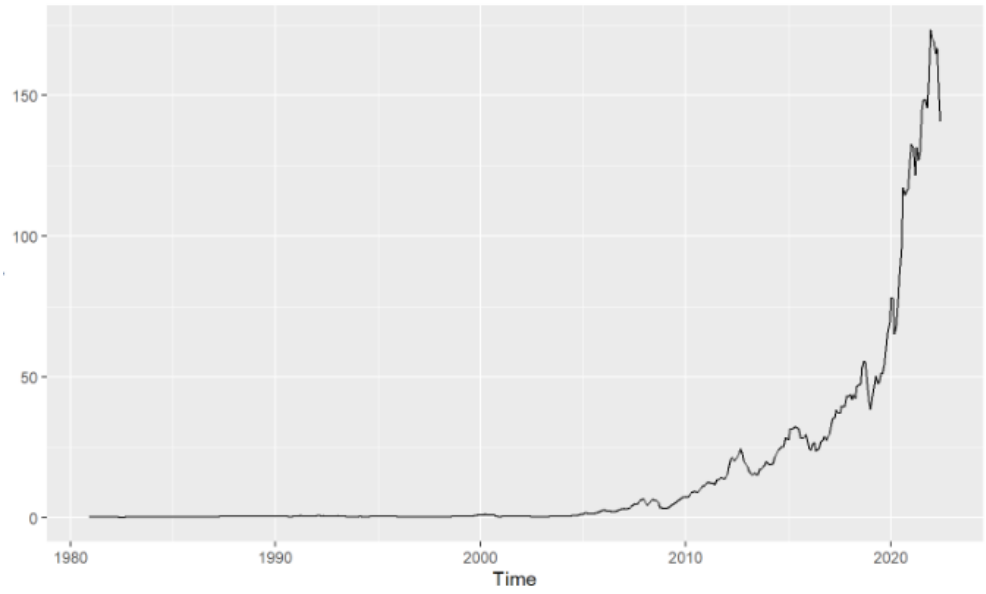
AAPL_ts <- ts(data = APPL_m$Close, start = c(1980,12,01), frequency = 12)

class(AAPL_ts)

#> [1] "ts"

We successfuly make our data APPL_m to ts (time series).

Plot AAPL_ts



STEP 6: CROSS VALIDATION

Cross Validation by splitting Data Train and Data Test.

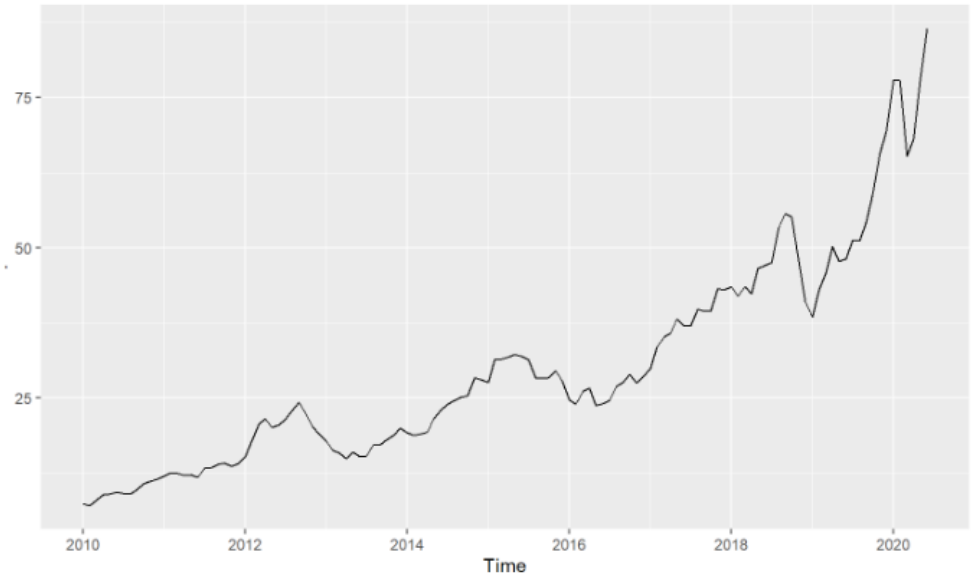
 # test

AAPL_test <- tail(AAPL_w, 24)

 # train

AAPL_train <- head(AAPL_w, -length(AAPL_test))

Check AAPL_train with visualization



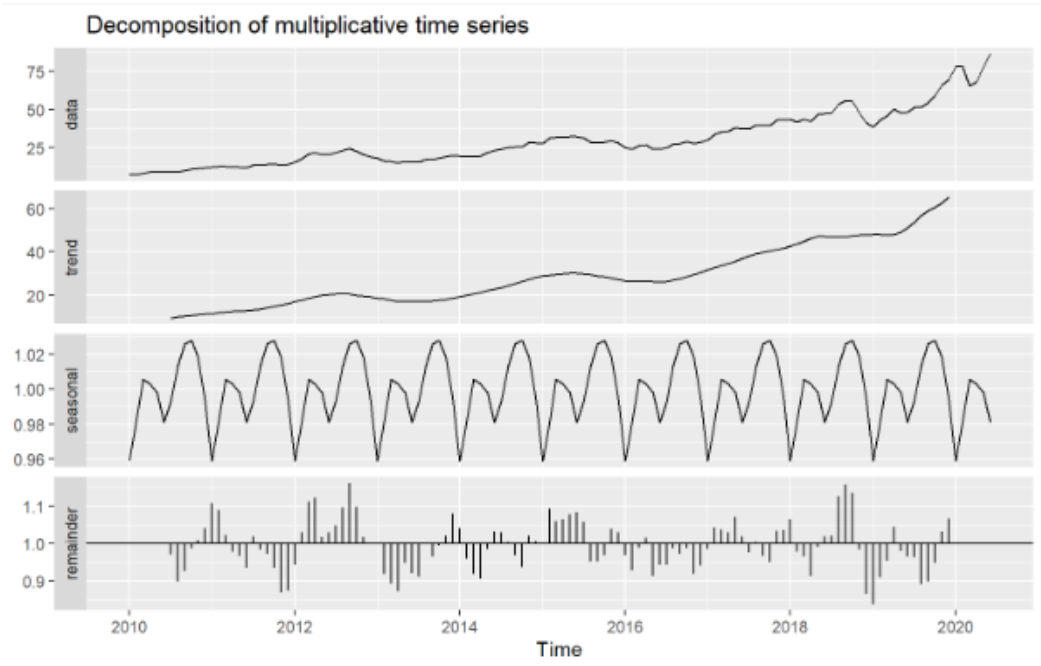
The graph above show AAPL_train already splitted. As we can see the time (x-axis) has less observation than before.

STEP 7: CHECK AAPL_TRAIN TREND AND SEASONAL WITH DECOMPOSE

AAPL_dc <- decompose(AAPL_train, type = "multiplicative")

AAPL_dc %>% autoplot()

> AAPL_train, This data has an increasing trend from time to time. And also has seasonal. So, we use Hold Winters method.



AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT:

Thus the apply analytics for forecasting and inventory planning for a large retailer is implemented successfully in R studio.

Ex.No:6	PERFORM PREDICTIVE ANALYTICS FOR CUSTOMERS' BEHAVIOUR INMARKETING AND SALES

AIM:

To perform predictive analytics for customers' behaviour in marketing and sales

ALGORITHM:

Step 1: Import the data into R Studio and clean it to remove any outliers or missing values.

Step 2: Transform the data into a format suitable for predictive modeling.

Step 3: Exploratory Data Analysis: Identify correlations between different variables.

Step 4: Feature Selection: Select the most relevant variables to include in the predictive model.

Step 5: Predictive Modelling: Build a predictive model using algorithms such as logistic regression, decision trees, or random forests.

Step 6: Evaluate the model's performance on the test data.

PROCEDURE:

STEP 1: IMPORT THE DATA INTO R STUDIO AND CLEAN IT TO REMOVE ANY OUTLIERS OR MISSING VALUES.

```
library(caret)

customer_data <- read.csv("customer_data.csv")

customer_data_clean <- na.omit(customer_data)
```

STEP 2: TRANSFORM THE DATA INTO A FORMAT SUITABLE FOR PREDICTIVE MODELING.

```
set.seed(1234)

train_index <- createDataPartition(y = customer_data_clean$Bought_Product,
                                   p = 0.7, list = FALSE)

train_data <- customer_data_clean[train_index,]

test_data <- customer_data_clean[-train_index,]
```

STEP 3: EXPLORATORY DATA ANALYSIS: IDENTIFY CORRELATIONS BETWEEN DIFFERENT VARIABLES.

```
summary(train_data)

ggplot(train_data, aes(x = Age, y = Income, color = Bought_Product)) +

  geom_point()
```

STEP 4: FEATURE SELECTION: SELECT THE MOST RELEVANT VARIABLES TO INCLUDE IN THE PREDICTIVE MODEL.

```
cor_matrix <- cor(train_data[,c("Age", "Gender", "Income", "Education", "Bought_Product")])

corrplot(cor_matrix)

# Remove variables that are highly correlated or not relevant

train_data_selected <- train_data[,c("Age", "Gender", "Income", "Bought_Product")]

test_data_selected <- test_data[,c("Age", "Gender", "Income", "Bought_Product")]
```

STEP 5: PREDICTIVE MODELLING: BUILD A PREDICTIVE MODEL USING ALGORITHMS SUCH AS LOGISTIC REGRESSION, DECISION TREES, OR RANDOM FORESTS.

```
model <- train(Bought_Product ~ ., data = train_data_selected, method = "glm", family = "binomial")

predictions <- predict(model, test_data_selected[,1:3], type = "prob")

confusionMatrix(predictions, test_data_selected$Bought_Product)
```

Step 6: Evaluate the model's performance on the test data.

```
new_customer <- data.frame(Age = 30, Gender = "Male", Income = 50000)

predict(model, new_customer, type = "prob")
```

AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT: Thus the perform predictive analytics for customers' behaviour inmarketing and sales is implemented successfully in R studio.

Ex.No:7	TIME SERIES DECOMPOSITION OF DATA

AIM:

To read and decompose dataet with time series analysis

ALGORITHM:

Step 1: Reading in Data

Step 2: Plot the difference in the seasons

Step 3: Time Series Decomposition

PROCEDURE:

install.packages("tidyverse")

install.packages("seasonal")

install.packages("fpp2")

library(tidyverse)

library(seasonal)

library(fpp2)

STEP 1: READING IN DATA

furniture <- read_csv("give file path with file name.csv") //example //D//store.csv

head(furniture)

	realtime_start	realtime_end	date	value
1	2018-12-11	2018-12-11	1992-01-01	3311
2	2018-12-11	2018-12-11	1992-02-01	3360
3	2018-12-11	2018-12-11	1992-03-01	3445
4	2018-12-11	2018-12-11	1992-04-01	3415
5	2018-12-11	2018-12-11	1992-05-01	3510
6	2018-12-11	2018-12-11	1992-06-01	3521

This is monthly data and that it begins in 1992. Now turn the value column into a time series object with the function ts() and then we will print it out.

> sales <- ts(furniture$value)

> sales

Time Series:

Start = 1

End = 321

Frequency = 1

[1] 3311 3360 3445 3415 3510 3521 3470 3461 3521 3576 3592 3579 3600 3636 3525 3672 3739 3721 3811

[20] 3815 3861 3837 3810 3761 3984 4037 4040 4100 4017 4066 4138 4179 4127 4185 4189 4223 4233 4153

[39] 4199 4274 4276 4340 4312 4397 4405 4404 4520 4476 4484 4557 4539 4569 4596 4626 4645 4573 4617

[58] 4732 4704 4766 4787 4807 4860 4856 4785 4890 4909 4952 4979 4915 4996 4818 4943 4959 5013 4946

[77] 5040 4958 4994 4981 5015 5073 5015 5073 5117 5144 5087 5185 5165 5165 5195 5150 5358 5344 5434

[96] 5350 5373 5284 5406 5220 5289 5293 5210 5236 5490 5191 5166 5317 4956 5020 5036 5130 5117 5144

> sales <- ts(furniture$value, frequency = 12, start = 1992)

> sales

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1992 3311 3360 3445 3415 3510 3521 3470 3461 3521 3576 3592 3579

1993 3600 3636 3525 3672 3739 3721 3811 3815 3861 3837 3810 3761

1994 3984 4037 4040 4100 4017 4066 4138 4179 4127 4185 4189 4223

1995 4233 4153 4199 4274 4276 4340 4312 4397 4405 4404 4520 4476

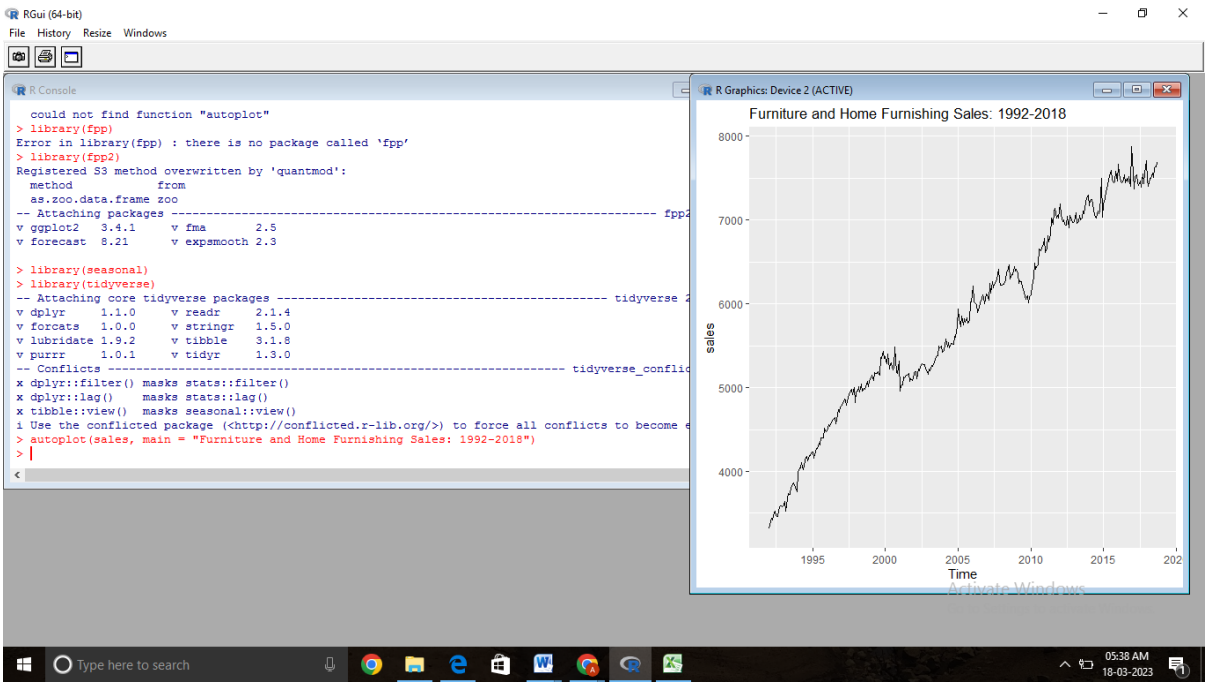
1996 4484 4557 4539 4569 4596 4626 4645 4573 4617 4732 4704 4766

1997 4787 4807 4860 4856 4785 4890 4909 4952 4979 4915 4996 4818

1998 4943 4959 5013 4946 5040 4958 4994 4981 5015 5073 5015 5073

1. Basic Plots
    - Time Plots, autoplot()
    - Season Plots, ggseasonplot()

autoplot(sales, main = "Furniture and Home Furnishing Sales: 1992-2018")



There is a general updward trend

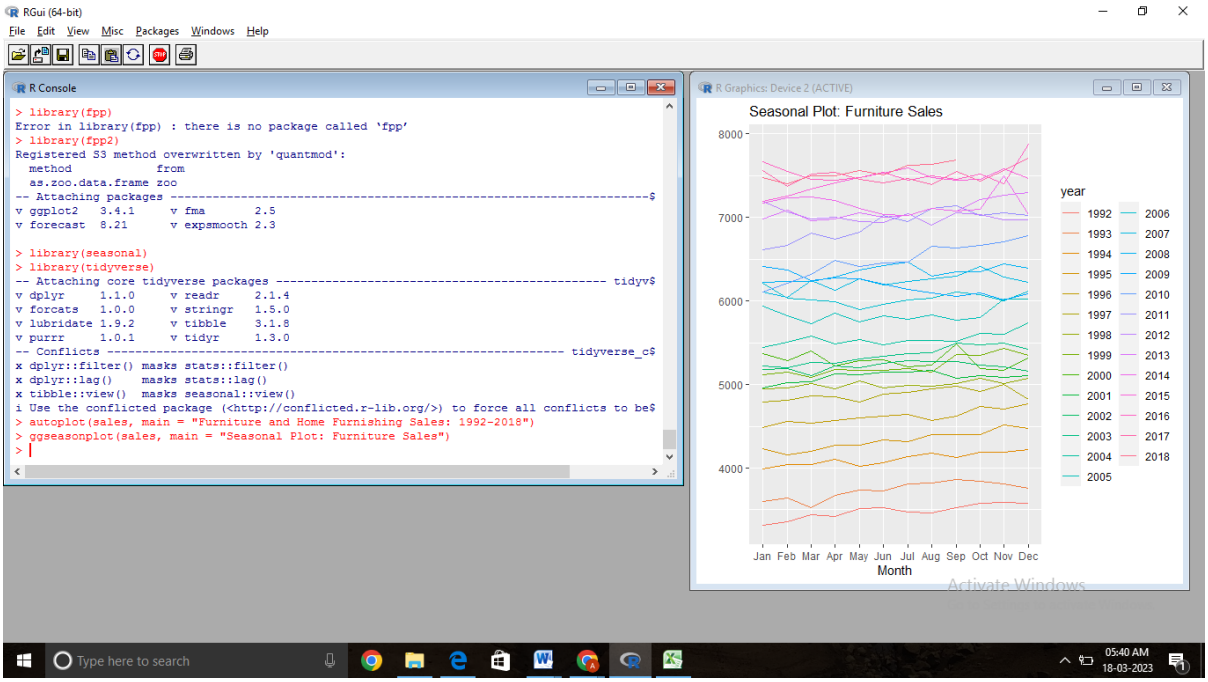
The trend is not constant, it moves down during the recession

There is differences in sales based on month

STEP 2: PLOT THE DIFFERENCE IN THE SEASONS

To get a clearer understanding of the differences in the seasons (the months) lets make a seasonplot with ggseasonplot()

ggseasonplot(sales, main = "Seasonal Plot: Furniture Sales")

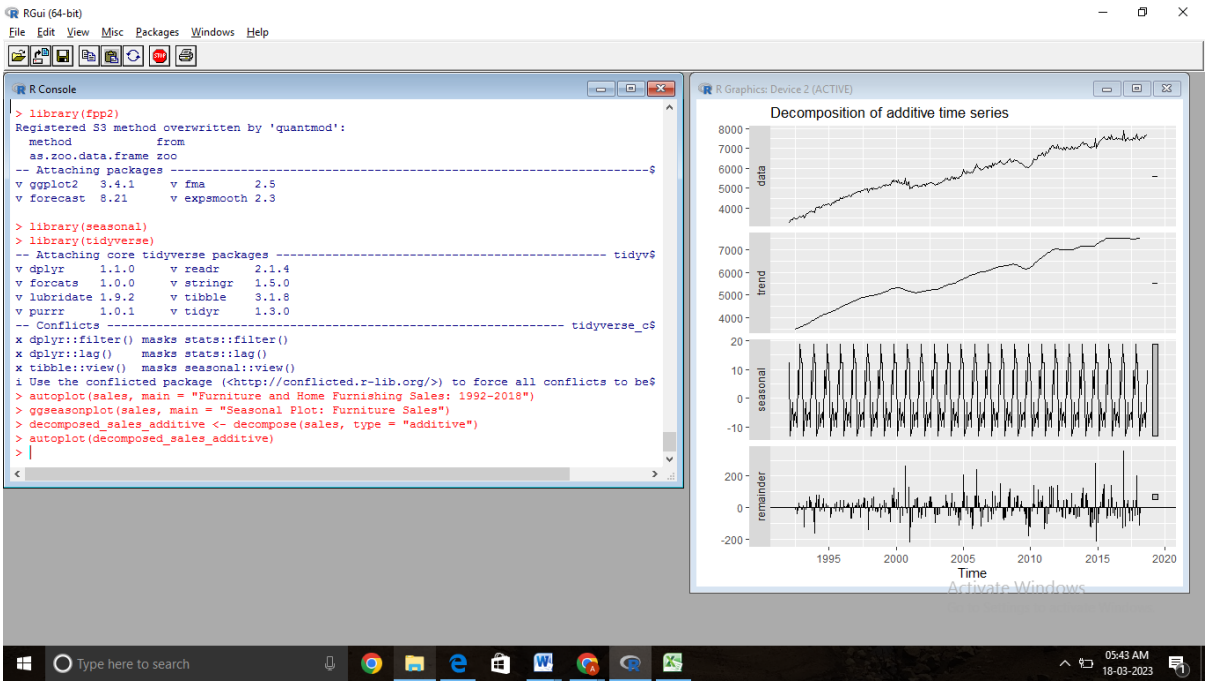


From this graph we can see that there is a jump at the end of each year, and that there is a small bump in March and August.

STEP 3: TIME SERIES DECOMPOSITION

decomposed_sales_multiplicative <- decompose(sales, type = "multiplicative")

autoplot(decomposed_sales_multiplicative)



Our original data (in panel 1, above) is now broken into 3 components.

a seasonal component (panel 2)

a trend component (panel 3)

a remained (panel 4)

Because our decomposition was additive, we can add the series in panels 2, 3, and 4 and get the top panel.

data = trend + seasonal + remainder

Here, remainder values of 0 are ideal, because that means the data can be explained by the season and the trend alone.

Remember that there are 12 seasonal values, one for each month that we add (or subtract) to the trend. These 12 seasonal values do not change over time and were found by averaging the seasonal affect of each month across all years.

> install.packages("pacman")

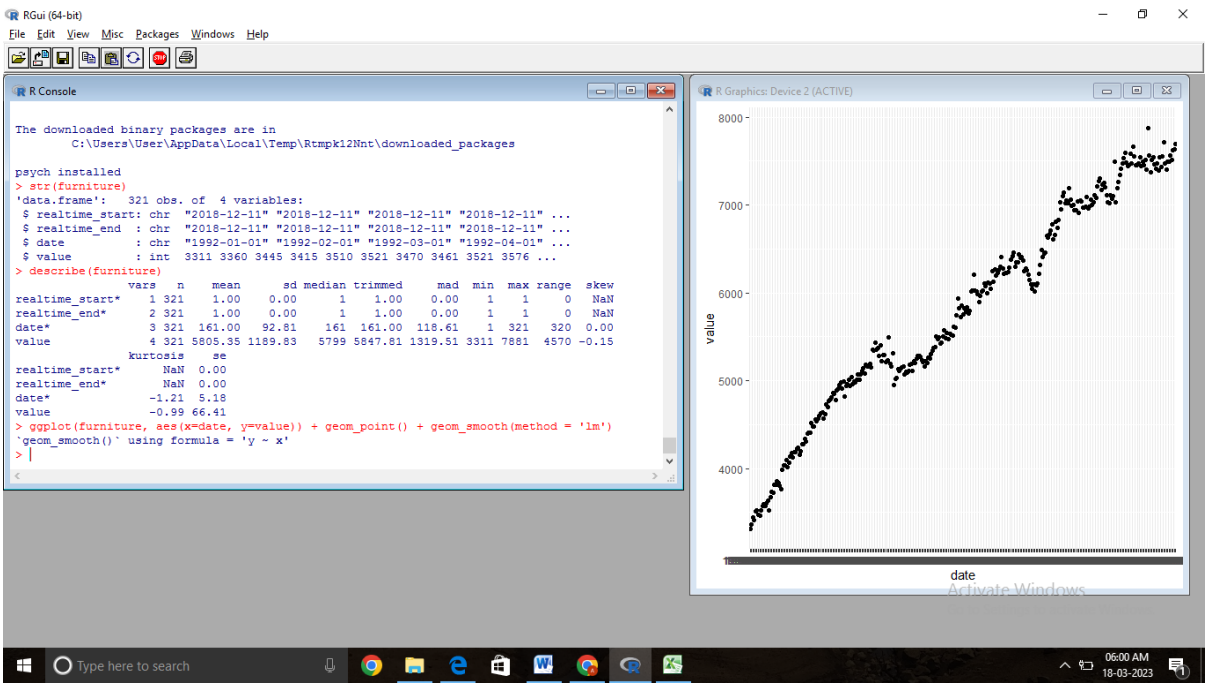
> library(pacman)

> pacman::p_load(pacman, tidyverse, openxlsx, forecast, psych)

> str(furniture)

> describe(furniture)

> ggplot(furniture, aes(x=date, y=value)) + geom_point() + geom_smooth(method = 'lm')



> furniture_ts <- ts(data = furniture[,c(2,3)])

> furniture_ts

Time Series:

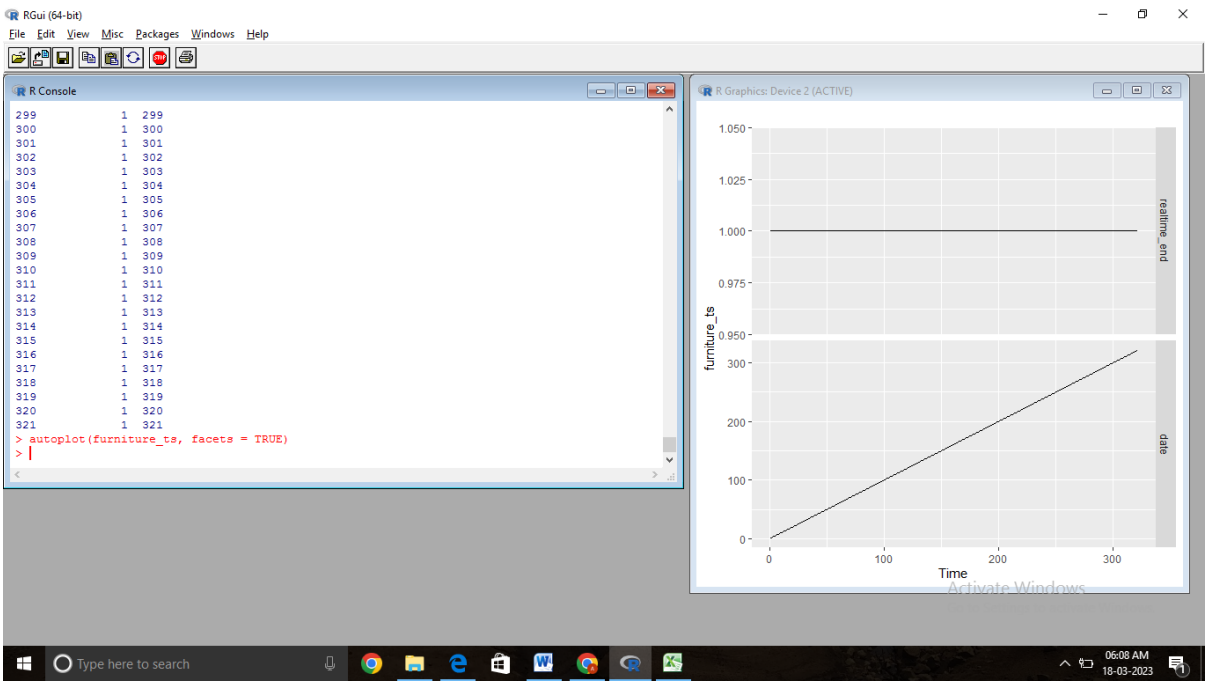
Start = 1

End = 321

Frequency = 1

	realtime_end	*date*
1	*1*	*1*
2	*1*	*2*
3	*1*	*3*
4	*1*	*4*
5	*1*	*5*
6	*1*	*6*
7	*1*	*7*

> autoplot(furniture_ts, facets = TRUE)



> fit <- auto.arima(furniture_ts[, "realtime_end"], xreg = furniture_ts[, "date"], stationary = TRUE)

> fit

Series: furniture_ts[, "realtime_end"]

ARIMA(0,0,0) with non-zero mean

Coefficients:

intercept

1

sigma^2 = 0: log likelihood = Inf

AIC=-Inf AICc=-Inf BIC=-Inf

> sales_increase <- coefficients(fit)[3]

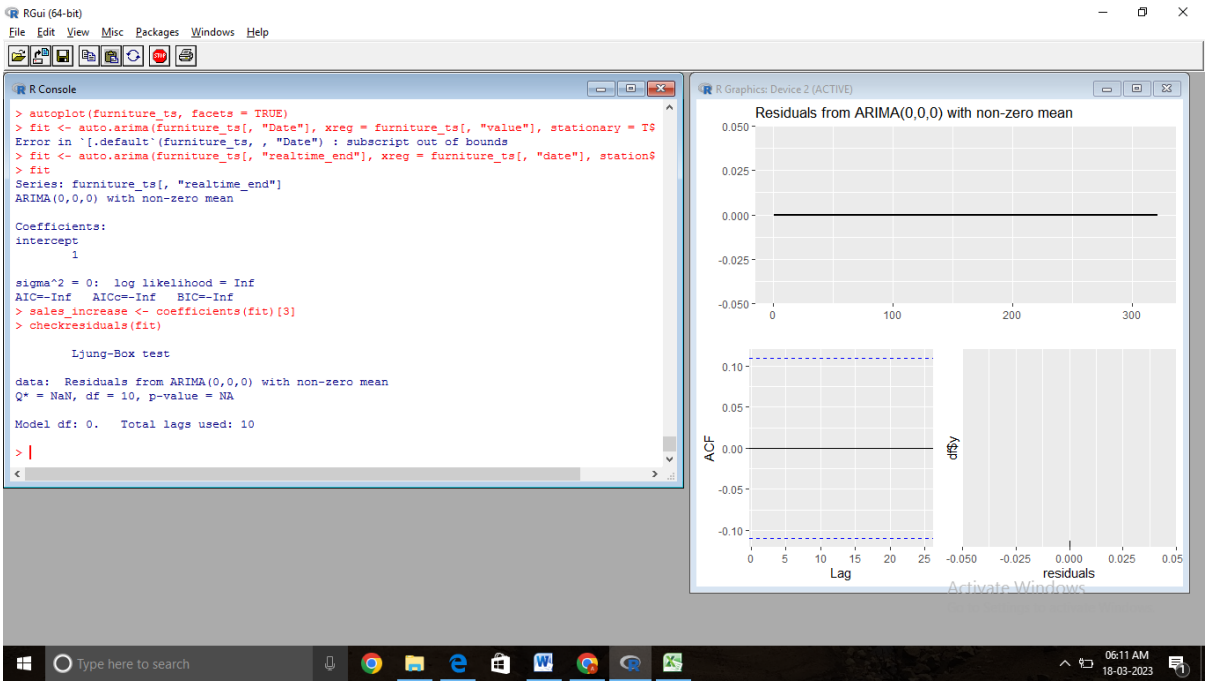
> checkresiduals(fit)

Ljung-Box test

data: Residuals from ARIMA(0,0,0) with non-zero mean

Q* = NaN, df = 10, p-value = NA

Model df: 0. Total lags used: 10



AIM & DATASET DESCRIPTION	/25
ALGORITHM	/20
PROCEDURE	/45
VIVA	/10
TOTAL	/100

RESULT:

Thus the time series decomposition of data is implemented successfully in R studio.