Roll No: 17E111

Ex. No: 1
Date : 01-07-2019

Data Manipulation in R

Aim:

To perform the basic datatype and arithmetic operations in R .

Procedure:

1. Download R for windows/ linux and RStudio from web and Install both the packages.
2. Open R studio.
3. In the script window perform all the assignment operations, basic data type operations and arithmetic operations.
4. After performing all the operations, Click on Save button to save the document in local disk.
5. Exit from R studio.

Vector:

Creat two vectors a,b
A is 1,2,3,4,5,6
B is 3,2,4,1,9

a)Find the result for the product of two vectors

```
a=c(1,2,4,5,6)
b=c(3,2,4,1,9)
a*b
```

```
> print(a*b)
[1]  3  4 16  5 54
```

b)Display the output of A and B row wise
rbind(a,b)

```
> print(rbind(a,b))
  [,1] [,2] [,3] [,4] [,5]
a    1    2    4    5    6
b    3    2    4    1    9
```

c)Display the output of A and B column wise
cbind(a,b)

```
> print(cbind(a,b))
     a b
[1,] 1 3
[2,] 2 2
[3,] 4 4
[4,] 5 1
[5,] 6 9
```

d)Check the output for A<=B
a<=b

```
> a<=b
[1]  TRUE  TRUE  TRUE FALSE  TRUE
```

e)Check the data type for A and B
print(paste("Vector a is",class(a),"and Vector b is",class(b)))

```
> print(paste("Vector a is",class(a),"and Vector b is",class(b)))
[1] "Vector a is numeric and Vector b is numeric"
```

Roll No: 17E111

Data Frame:
Create a vector with Age, Name and Gender with the following data Age <- c(22, 25, 18, 20) Name <- c("James", "Mathew", "Olivia", "Stella") Gender <- c("M", "M", "F", "F") From the above object create a table like structure and store in 'student_data'. Display the table and Print its corresponding data type.

```
Age=c(22, 25, 18, 20)
Name=c("James", "Mathew", "Olivia", "Stella")
Gender=c("M", "M", "F", "F")
student_data=data.frame(Age,Name,Gender)
print(student_data)
print(paste("Datatype is:",class(student_data$Gender)))
```

```
> print(student_data)
  Age   Name Gender
1  22  James      M
2  25 Mathew      M
3  18 Olivia      F
4  20 Stella      F
> print(paste("Datatype is:",class(student_data$Gender)))
[1] "Datatype is: factor"
```

Factors:
Create the vectors with height, weight and gender. height <- c(132,151,162,139,166,147,122) weight <- c(48,49,66,53,67,52,40) gender <- c("male","male","female","female","male","female","male") Create a dataframe with the above vectors and store it in 'input_data'. Test the gender column data type and print the result.

```
Height=c(132,151,162,139,166,147,122)
Weight=c(48,49,66,53,67,52,40)
Gender=c("male","male","female","female","male","female","male")
input_data=data.frame(Height,Weight,Gender)
print(input_data)
print(paste("Datatype is:",class(Gender)))
```

```
> print(input_data)
  Height Weight Gender
1    132     48   male
2    151     49   male
3    162     66 female
4    139     53 female
5    166     67   male
6    147     52 female
7    122     40   male
> print(paste("Datatype is:",class(Gender)))
[1] "Datatype is: character"
```

Array:
Create a 3D array with 5 rows and 4 columns. consider the data as a sample of 60 values with no replication.

```
res=array(sample(1:60,60,replace=TRUE),dim=c(5,4,3))
print(res)
```

Roll No: 17E111

```
> res=array(sample(1:60,60,replace=TRUE),dim=c(5,4,3))
> print(res)
, , 1

     [,1] [,2] [,3] [,4]
[1,]   44   33   55   36
[2,]    6    3   14   12
[3,]   15    6   24   39
[4,]   51    1   36   41
[5,]   44   37   25   18

, , 2

     [,1] [,2] [,3] [,4]
[1,]   27    2   47   12
[2,]   39   10   15   15
[3,]   30    9   23    1
[4,]   60   57   12   45
[5,]   46    1   32    6

, , 3

     [,1] [,2] [,3] [,4]
[1,]   35   16    6    4
[2,]   11   56   19   32
[3,]   49   34   10    2
[4,]   50   42    5   48
[5,]   24   38   47   44
```

Lists:

Create a list ‘data_list’ that contains a) Vector with Jan, Feb and Mar. b) a matrix with 2 rows and 3 column of 1, 2, 3, 4, -1, 9. c) list containing a character and a number of “Red” and 12.3.

```
vtr=c("jan","Feb","Mar")
s=matrix(c(1,2,3,4,-1,9),nrow=2,ncol=3)
data_list=list(vtr,s,"a","Red",12.3)
data_list
```

```
> print(data_list)
[[1]]
[1] "Jan" "Feb" "Mar"

[[2]]
     [,1] [,2] [,3]
[1,]    1    3   -1
[2,]    2    4    9

[[3]]
[[3]][[1]]
[1] "red"

[[3]][[2]]
[1] 12.3
```

Roll No: 17E111

Grep:
Create an object emails with the following mail id
john.doe@ivyleague.edu
education@world.gov
dalai.lama@peace.org
invalid.edu
quant@bigdatacollege.edu
cookie.monster@sesame.tv
1. Find the mail-id with ‘edu’.
2. Store in object ‘hits’ with all the mail id contains ‘edu’.
3. Use the variable hits to select from the emails vector only the emails that contain "edu”.
4. Use grepl() with the more advanced regular expression to return a logical vector of right mail-id with ‘.edu’ Simply print the result.
5. create a vector of indices. Store the result in the variable hits. (Same as step 2 and 3).
6. Replace the mail id’s ending with ‘.edu’ as ‘karpagam.edu’.

```
emails=c('john.doe@ivyleague.edu','education@world.gov','dalai.lama@peace.org',
        'invalid.edu','quant@bigdatacollege.edu','cookie.monster@sesame.tv')
hits=grep('edu',emails)
emails[hits]
hits=grepl('\\.edu',emails)
emails[hits]=sub('\\.edu','karpagam.edu',emails[hits])
```

```
> emails[hits]=sub('\\.edu','karpagam.edu',emails[hits])
> emails
[1] "john.doe@ivyleaguekarpagam.edu"   "education@world.gov"
[3] "dalai.lama@peace.org"             "invalidkarpagam.edu"
[5] "quant@bigdatacollegekarpagam.edu" "cookie.monster@sesame.tv"
```

Apply Function:
Create a dataframe ‘bmi’ with following vectors
Age<-c(56,34,67,33,25,28)
Weight<-c(78,67,56,44,56,89)
Height<-c(165, 171,167,167,166,181)
a) Find the row-wise sum of bmi
b) Find the Column wise mean of bmi
c) Calculate the BMI for each person. (Formula BMI = Weight (KG)/ Height^2 (M)

```
Age<-c(56,34,67,33,25,28)
Weight<-c(78,67,56,44,56,89)
Height<-c(165, 171,167,167,166,181)
bmi=data.frame(Age,Weight,Height)
rowMeans(bmi)
colMeans(bmi)
BMI=(bmi[2]/bmi[3]^2)*10000
```

```
> BMI=(bmi[2]/bmi[3]^2)*10000
> BMI
    Weight
1 28.65014
2 22.91303
3 20.07960
4 15.77683
5 20.32225
6 27.16645
```

Roll No: 17E111

Lapply and Sapply:
Create a list 'list1' with a as 1:10 and b as 11:20. Display the result of lapply and sapply.
list1=list(a=1:10,b=11:20)
lapply(list1,FUN=sum)
sapply(list1,FUN=sum)

```
> list1=list(a=1:10,b=11:20)
> lapply(list1,FUN=sum)
$a
[1] 55

$b
[1] 155

> sapply(list1,FUN=sum)
  a   b
 55 155
```

Tapply:
Load the 'iris' dataset from r. Find the median of 'Sepal.Width' of each 'Species'.

tapply(iris$Sepal.Width,FUN=median,INDEX = iris$Species)

```
> tapply(iris$Sepal.Width,FUN=median,INDEX = iris$Species)
    setosa versicolor  virginica
       3.4        2.8        3.0
```

Result:

Thus, all the datatype of basic operations of R has completed successfully.

Roll No: 17E111

Ex. No: 2

Data Import in R

Date: 15-07-2019

Aim:

To import datasets and perform the necessary operations in R.

Procedure:

1. Open R studio.
2. In the script window import all the files which is needed.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

1. Import "Sample.csv" file and Store in "SampData" object. Display First 6 and last 6 records from "SampData" object.

SampData=read.csv(file='C:/Users/gowri/Desktop/R Programming/Lab Data Set/Sample.csv')
head(SampData)
tail(SampData)

```
> head(SampData)
  X1   Eldon.Base.for.stackable.storage.shelf..platinum Muhammed.MacIntyre  X3 X.213.25 X38.94   X35 Nunavut         Storage...Organization X0.8
1  2 1.7 Cubic Foot Compact "Cube" Office Refrigerators       Barry French 293   457.81 208.16 68.02 Nunavut                     Appliances 0.58
2  3   Cardinal Slant-D® Ring Binder, Heavy Gauge Vinyl       Barry French 293    46.71   8.69  2.99 Nunavut Binders and Binder Accessories 0.39
3  4                                               R380      Clay Rozendal 483  1198.97 195.99  3.99 Nunavut   Telephones and Communication 0.58
4  5                              Holmes HEPA Air Purifier      Carlos Soltero 515    30.94  21.78  5.94 Nunavut                     Appliances 0.50
5  6  G.E. Longer-Life Indoor Recessed Floodlight Bulbs     Carlos Soltero 515     4.43   6.64  4.95 Nunavut              Office Furnishings 0.37
6  7  Angle-D Binders with Locking Rings, Label Holders       Carl Jackson 613   -54.04   7.30  7.72 Nunavut Binders and Binder Accessories 0.38
> tail(SampData)
    X1 Eldon.Base.for.stackable.storage.shelf..platinum Muhammed.MacIntyre    X3 X.213.25 X38.94   X35               Nunavut
94  95                  Bevis Boat-Shaped Conference Table      Doug Bickford 10499    31.21 262.11 62.74 Northwest Territories
95  96               Linden® 12" Wall Clock With Oak Frame      Doug Bickford 10535   -44.14  33.98 19.99 Northwest Territories
96  97                                         Newell 326      Doug Bickford 10535    -0.79   1.76  0.70 Northwest Territories
97  98                          Prismacolor Color Pencil Set      Jamie Kunitz 10789    76.42  19.84  4.10 Northwest Territories
98  99                            Xerox Blank Computer Paper    Anthony Johnson 10791    93.36  19.98  5.77 Northwest Territories
99 100                                      600 Series Flip      Ralph Knight 10945     4.22  95.99  8.99 Northwest Territories
          Storage...Organization X0.8
94                        Tables 0.75
95            Office Furnishings 0.55
96           Pens & Art Supplies 0.56
97           Pens & Art Supplies 0.44
98                         Paper 0.38
99 Telephones and Communication 0.57
```

2. Import "LungCap.csv" file and Store in "LungDetail" object. Summarize the "LungDetail"

LungDetail=read.csv2(file='C:/Users/gowri/Desktop/R Programming/Lab Data Set/LungCap.csv')
summary(LungDetail)

```
> summary(LungDetail)
    LungCap          Age               Height      Smoke       Gender      Caesarean
 8.35   :  8   Min.   : 3.00   65.4    :  8   no :648   female:358   no :561
 6.45   :  7   1st Qu.: 9.00   63.3    :  7   yes: 77   male  :367   yes:164
 7.825  :  7   Median :13.00   65.5    :  7
 8      :  7   Mean   :12.33   67.5    :  7
 8.775  :  7   3rd Qu.:15.00   69.3    :  7
 7.55   :  6   Max.   :19.00   73.5    :  7
 (Other):683                   (Other):682
```

3. Import "LungCap.txt" file and Store in "LungData" object. Display the average of *Lung Capacity* based on "Gender" Category.

LungData=read.table(file='C:/Users/gowri/Desktop/R Programming/Lab Data Set/LungCap.txt',header = T)
LungData=as.data.frame(LungData)
tapply(X=LungData$LungCap,INDEX=LungData$Gender,FUN=mean)

```
> tapply(X=LungData$LungCap,INDEX=LungData$Gender,FUN=mean)
  female     male
7.405746 8.309332
```

Roll No: 17E111

4. Import “NamelistFWF.txt” file and Store in “Namelist” object with the headers of Rollno, Branch, Name, Gender and Classification.

```
Namelist=read.fwf(file='C:/Users/gowri/Desktop/R Programming/Lab Data Set/NamelistFWF.txt',
            widths=c(6,3,15,1,-1,4),
            col.names=c('Rollno','Branch','Name','Gender','Classification'))
Namelist
```

```
> Namelist
   Rollno Branch            Name Gender Classification
1  17L115    ECE GOPINATH A          M           PASS
2  17L138    ECE ROGITH S            M           PASS
3  17L240    ECE SARAVANAN S         M           PASS
4  17L241    ECE SHOBIKA S           F           PASS
5  17L310    ECE DHIVYA BHARATHI     F           PASS
6  17L327    ECE MOHAMMED SHAM       M           PASS
7  17F101    I.T ABISHAK R           F           PASS
8  17F213    I.T KANAGA LAKS R       F           PASS
9  17T127    I.T PREETHI K           F           PASS
10 17P106    CSE ARUNPRASATH P       M           PASS
11 17P206    CSE BALACHANDRU S       M           PASS
12 17N106    EIE ARAVINTH V          M           PASS
13 17N128    EIE PRIYADHARSHINI      F           PASS
14 17N231    EIE SARAVANAN P         M           PASS
15 17E111    EEE J GOWRI SANKAR      M           PASS
16 17E138    EEE SRIRAM R            M           PASS
17 17E203    ANU JA B                F           PASS
18 17E225    MAN IKANDA PRABHU       M           PASS
19 17E302    AKS HAYAMATHI P         F           PASS
20 17E331    PRI YADHARSHINI S       F           PASS
```

5. Import the data “crime.sav”, Fetch and display all information of Murder, Auto and State. Extract all the information where the rate of murder is greater than 13 and store in ‘crime.txt’.

```
library(foreign)
crimeData=read.spss('C:/Users/gowri/Desktop/R Programming/Lab Data Set/crime.sav',to.data.frame = T)
df=data.frame(crimeData$STATE,crimeData$MURDER,crimeData$AUTO)
df[df[,2]>13,]
```

```
> df[df[,2]>13,]
   crimeData.STATE crimeData.MURDER crimeData.AUTO
1   ALABAMA                    14.2          280.7
18  LOUISIANA                  15.5          337.7
24  MISSISSIPPI                14.3          144.4
28  NEVADA                     15.8          559.2
43  TEXAS                      13.3          397.6
```

6. Load and display the Customer First Name starts with ‘A’ from CustomerDB.mdb file and store in ‘mdbtostat.sav’.

```
library(RODBC)
dbCon=odbcDriverConnect("Driver={Microsoft Access Driver (*.mdb, *.accdb)};
              DBQ=C:/Users/gowri/Desktop/R Programming/Lab Data Set/CustomerDB.mdb")
fn=sqlQuery(channel = dbCon,query = "select FirstName from customerT")
grep('^A',fn)
write.spss(x=,file='mdbtostat.sav')
```

7. Load the data between “1990-2000” from ‘urbanpop’ and write in ‘urbanpop.csv’.

```
library(readxl)
excel_sheets('C:/Users/gowri/Desktop/R Programming/Dataset/urbanpop.xls')
up=read_xls('C:/Users/gowri/Desktop/R Programming/Dataset/urbanpop.xls',sheet="1975-2011")
up1=up[,c(1,17:27)]
```

Roll No: 17E111

write.csv(up1,file='urbanpop.csv')

```
> library(readxl)
> excel_sheets('C:/Users/gowri/Desktop/R Programming/Dataset/urbanpop.xls')
[1] "1960-1966" "1967-1974" "1975-2011"
> up=read_xls('C:/Users/gowri/Desktop/R Programming/Dataset/urbanpop.xls',sheet="1975-2011")
> up1=up[,c(1,17:27)]
> write.csv(up1,file='urbanpop.csv')
```

	country	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000
1	Afghanista	3454129	3617842	3788685	3966956	4152960	4347018	4531285	4722603	4921227	5127421	5341456
2	Albania	1198293	1215445	1222544	1222812	1221364	1222234	1228760	1238090	1250366	1265195	1282223
3	Algeria	13177079	13708813	14248297	14789176	15322651	15842442	16395553	16935451	17469200	18007937	18560597
4	American S	38087.65	39599.81	41049.36	42442.81	43798	45129.32	46342.57	47527.04	48705.3	49905.95	51151.41
5	Andorra	49982.36	51972.21	54469.43	57079.19	59243.05	60597.73	60927.11	60461.99	59685.43	59280.95	59718.58
6	Angola	3838852	4102967	4388137	4690892	5004756	5324794	5602207	5882843	6173329	6483827	6822112
7	Antigua an	22035.68	22047.36	22228.62	22537.77	22914.21	23311.51	23667.11	24030.72	24379.28	24690.41	24948.54

Result:

Thus, all the datatype of basic operations of R has completed successfully.

Roll No: 17E111

Ex. No: 3
Date: 05-08-2019

Exploratory Data Analysis in R

Aim:

To perform Exploratory Data Analysis (EDA) operations for categorical and numerical datasets in R.

Procedure:

1. Open R studio.
2. In the script window perform all the EDA operations for categorical and numerical data.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

EDA - Categorical

comics=read.csv('C:/Users/gowri/Desktop/Clg/R Programming/rprgmn/comics.csv')
glimpse(comics)

```
> glimpse(comics)
Observations: 23,272
Variables: 11
$ name         <fct> Spider-Man (Peter Parker), Captain America (Steven Rogers), Wolverine (James ...
$ id           <fct> Secret, Public, Public, Public, No Dual, Public, Public, Public, Public, Publ...
$ align        <fct> Good, Good, Neutral, Good, Good, Good, Good, Good, Neutral, Good, Good, Good,...
$ eye          <fct> Hazel Eyes, Blue Eyes, Blue Eyes, Blue Eyes, Blue Eyes, Blue Eyes, Brown Eyes...
$ hair         <fct> Brown Hair, White Hair, Black Hair, Black Hair, Blond Hair, No Hair, Brown Ha...
$ gender       <fct> Male, Male, Male, Male, Male, Male, Male, Male, Male, Male, Female, Mal...
$ gsm          <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ alive        <fct> Living Characters, Living Characters, Living Characters, Living Characters, L...
$ appearances  <int> 4043, 3360, 3061, 2961, 2258, 2255, 2072, 2017, 1955, 1934, 1825, 1713, 1528,...
$ first_appear <fct> Aug-62, Mar-41, Oct-74, Mar-63, Nov-50, Nov-61, Nov-61, May-62, Sep-63, Nov-6...
$ publisher    <fct> marvel, marvel, marvel, marvel, marvel, marvel, marvel, marvel, marvel, marve...
```

names(comics)

```
> names(comics)
 [1] "name"         "id"           "align"        "eye"          "hair"         "gender"
 [7] "gsm"          "alive"        "appearances"  "first_appear" "publisher"
```

levels(comics$align)

```
> levels(comics$align)
[1] "Bad"                "Good"               "Neutral"            "Reformed Criminals"
```

table(comics$align,comics$gender)#[-4,]

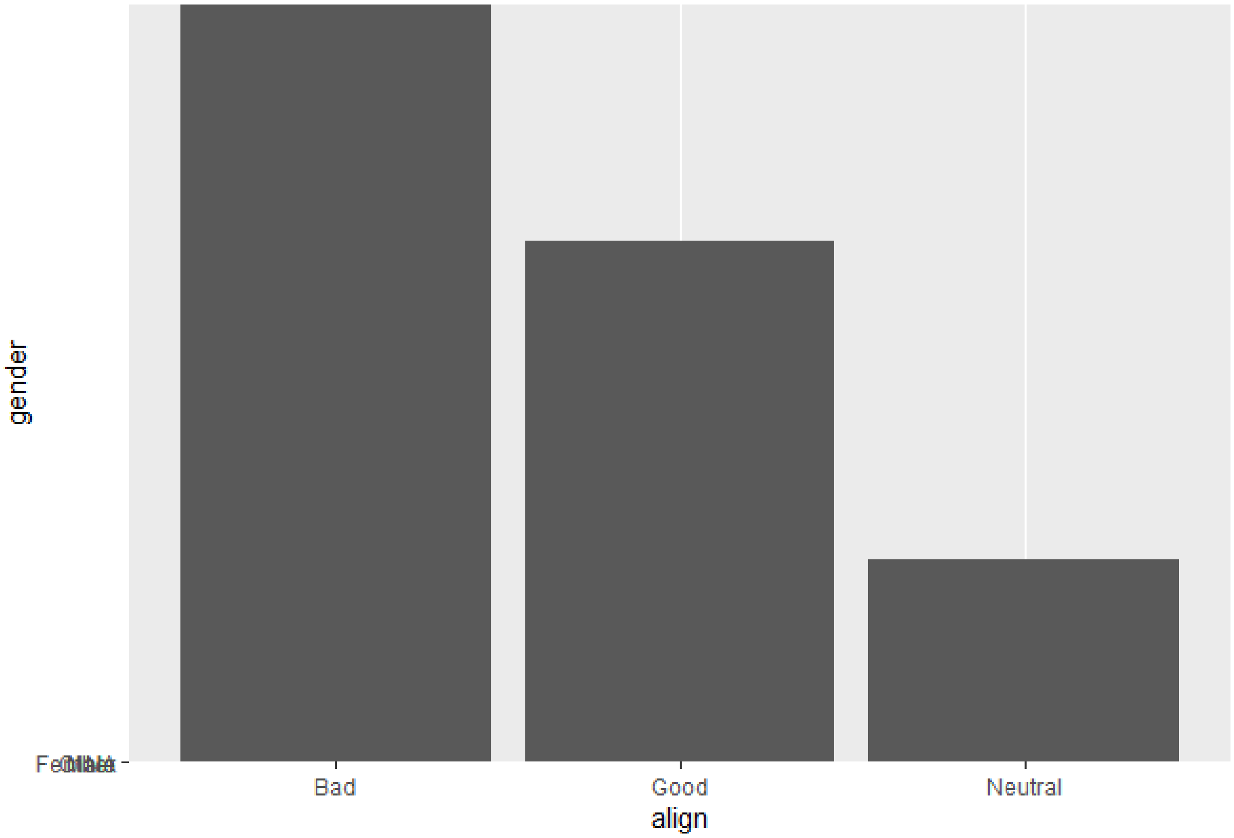
```
> table(comics$align,comics$gender)#[-4,]

                     Female Male Other
  Bad                  1573 7561    32
  Good                 2490 4809    17
  Neutral               836 1799    17
  Reformed Criminals      1    2     0
```

comics=comics %>% filter(align!="Reformed Criminals") %>% droplevels()
levels(comics$align)

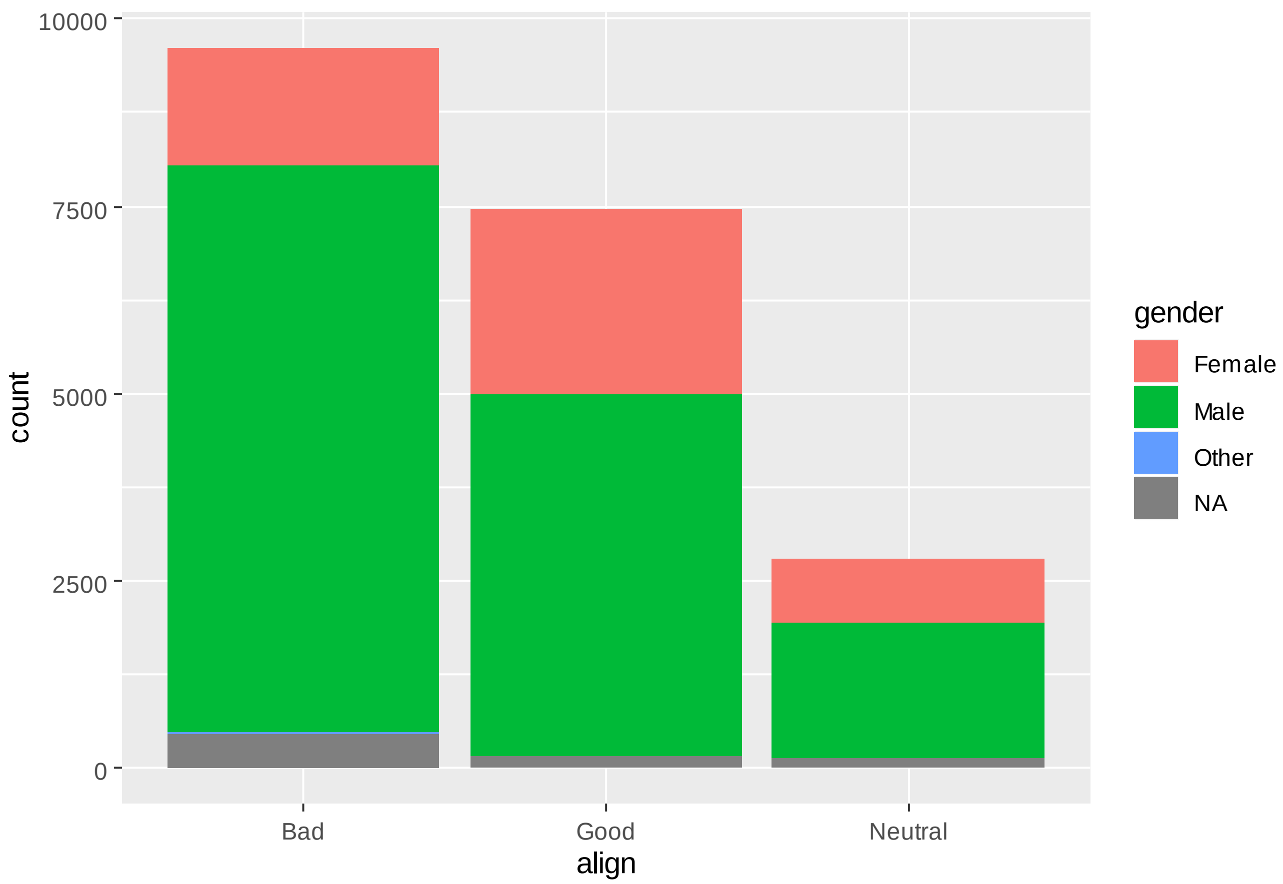
```
> levels(comics$align)
[1] "Bad"     "Good"    "Neutral"
```

Roll No: 17E111

ggplot(comics,aes(x=align,y=gender))+geom_col()

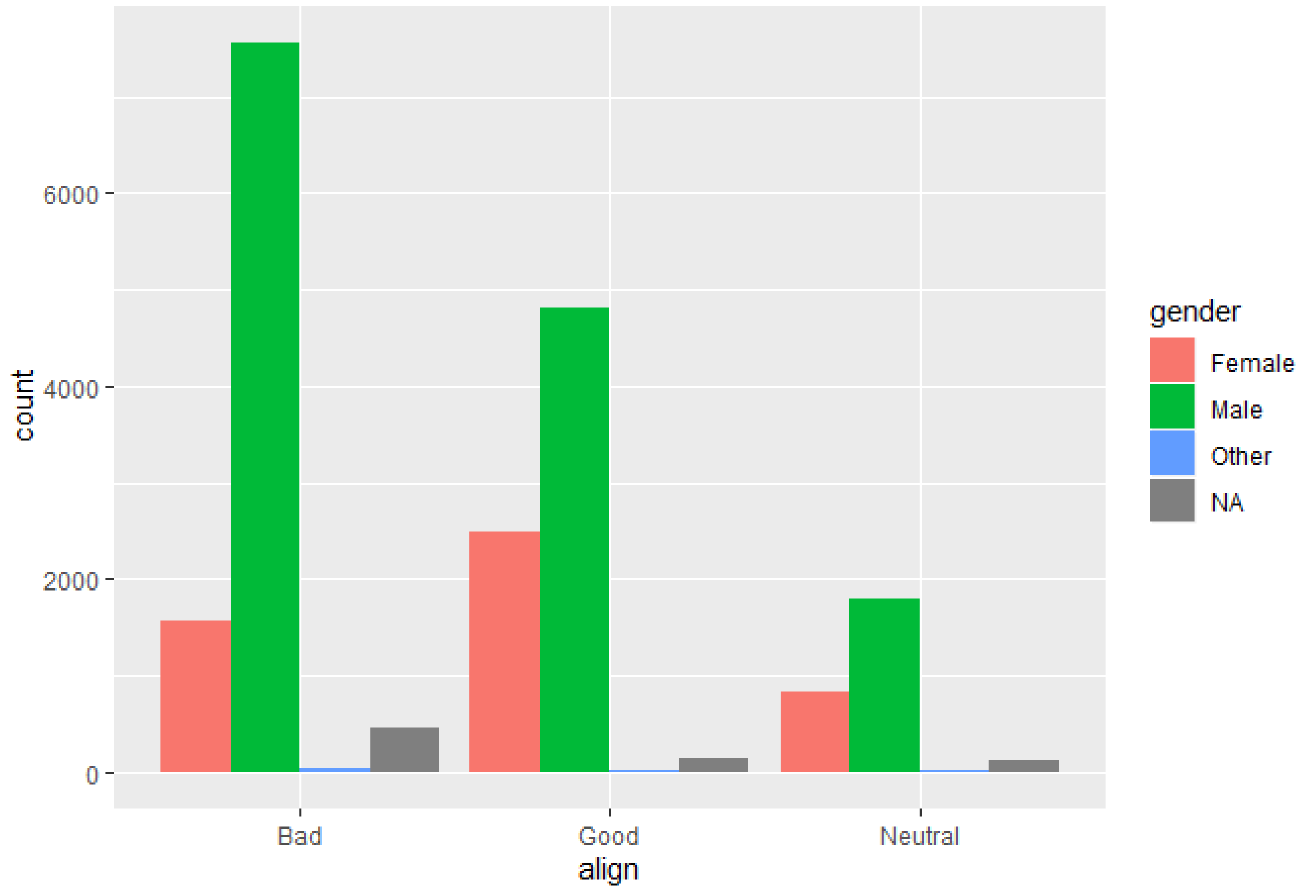


ggplot(comics,aes(x=align,fill=gender))+geom_bar()

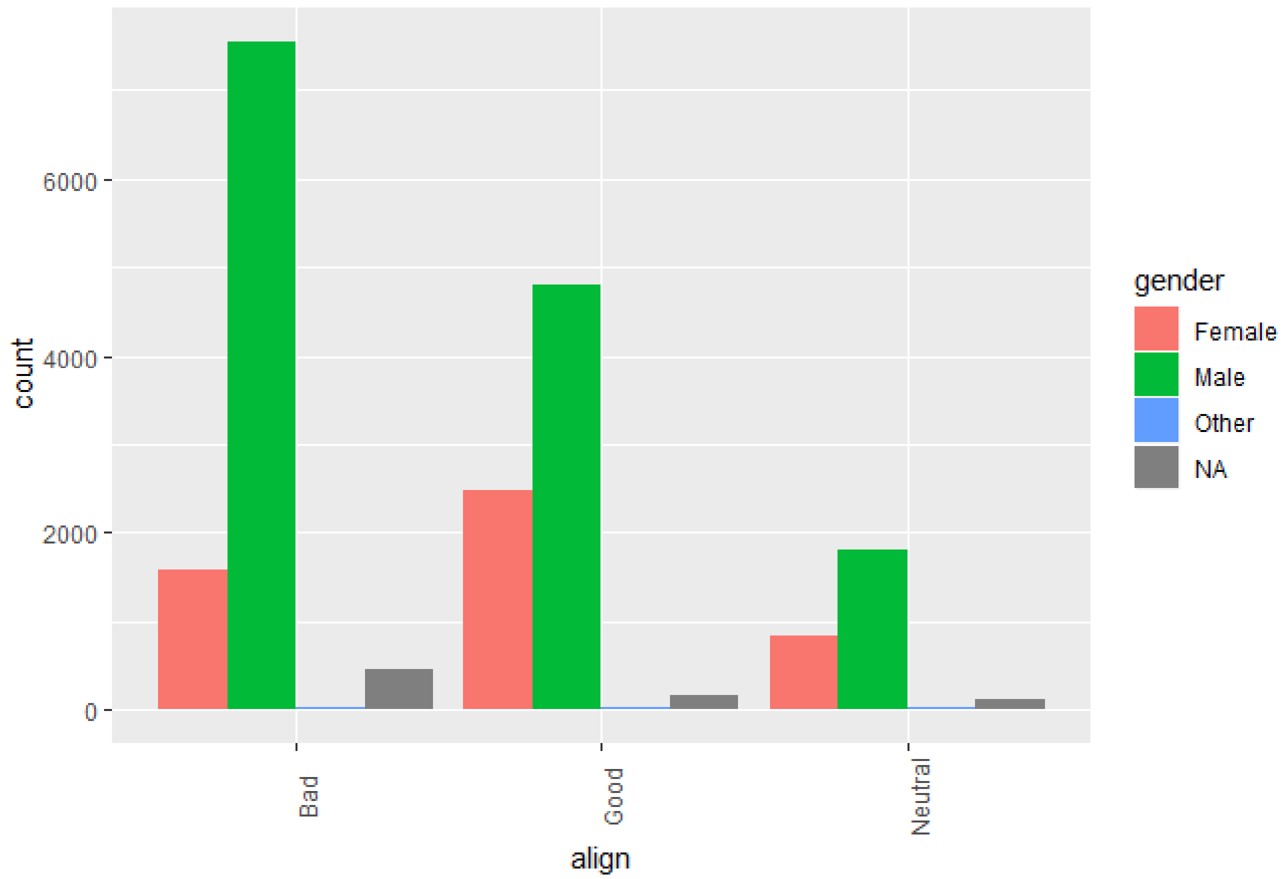


Roll No: 17E111

ggplot(comics,aes(x=align,fill=gender))+geom_bar(position = 'dodge')

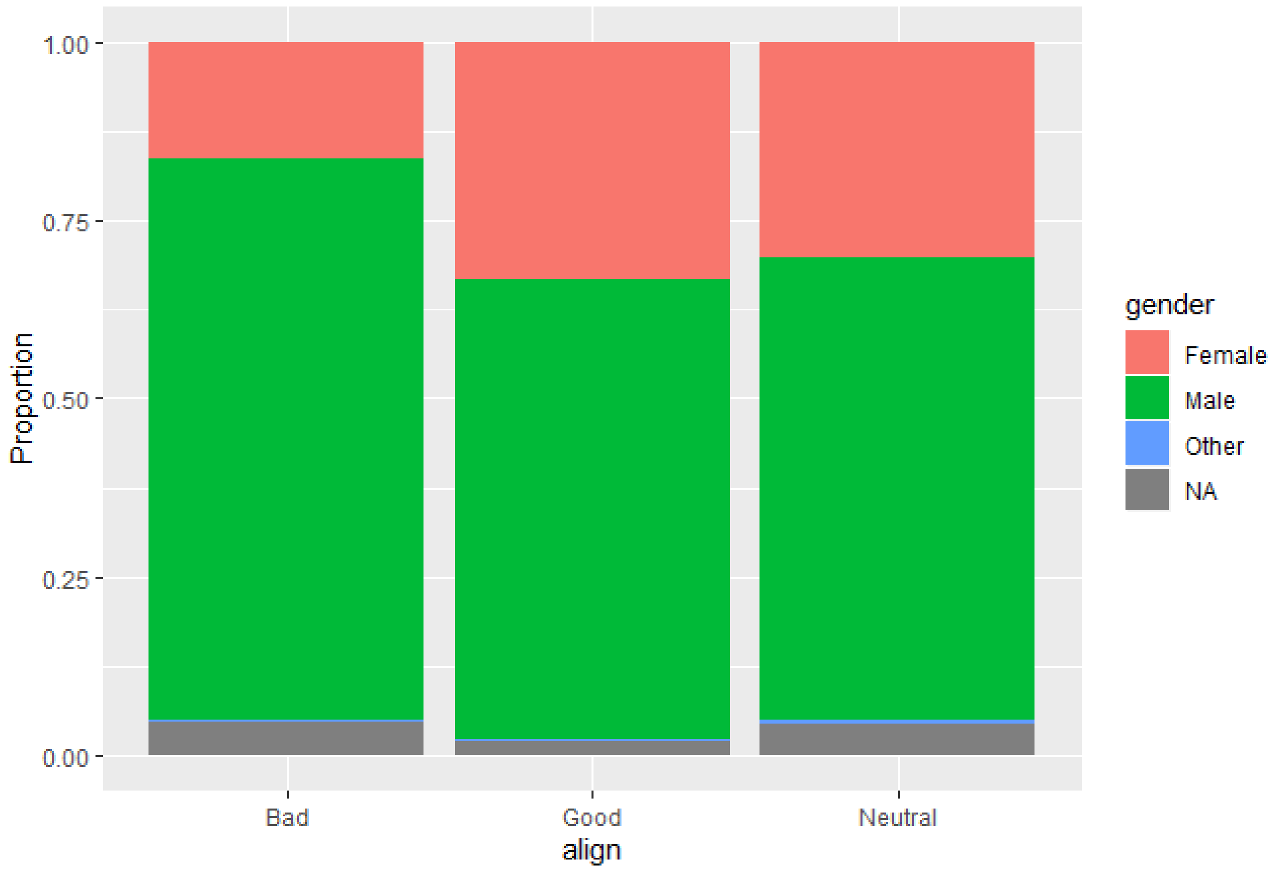


ggplot(comics,aes(x=align,fill=gender))+geom_bar(position = 'dodge')+theme(axis.text.x=element_text(angle = 90))

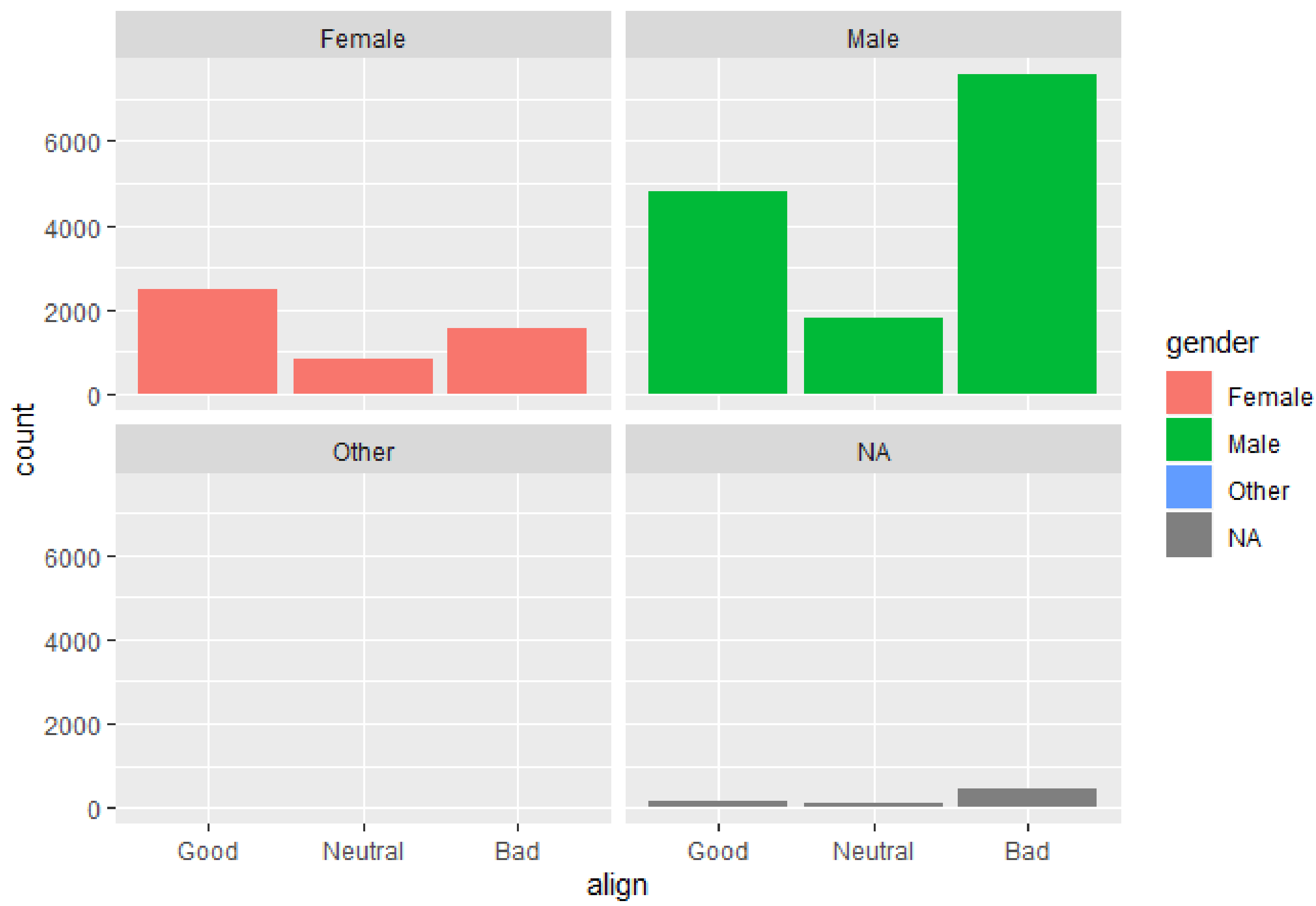


Roll No: 17E111

```
ggplot(comics,aes(x=align,fill=gender))+geom_bar(position = 'fill')+ylab("Proportion")
```




```
comics$align=factor(comics$align,levels = c('Good','Neutral','Bad'))
ggplot(comics,aes(x=align,fill=gender))+geom_bar(position = 'dodge')+facet_wrap(~gender)
```



Roll No: 17E111

EDA – Numerical

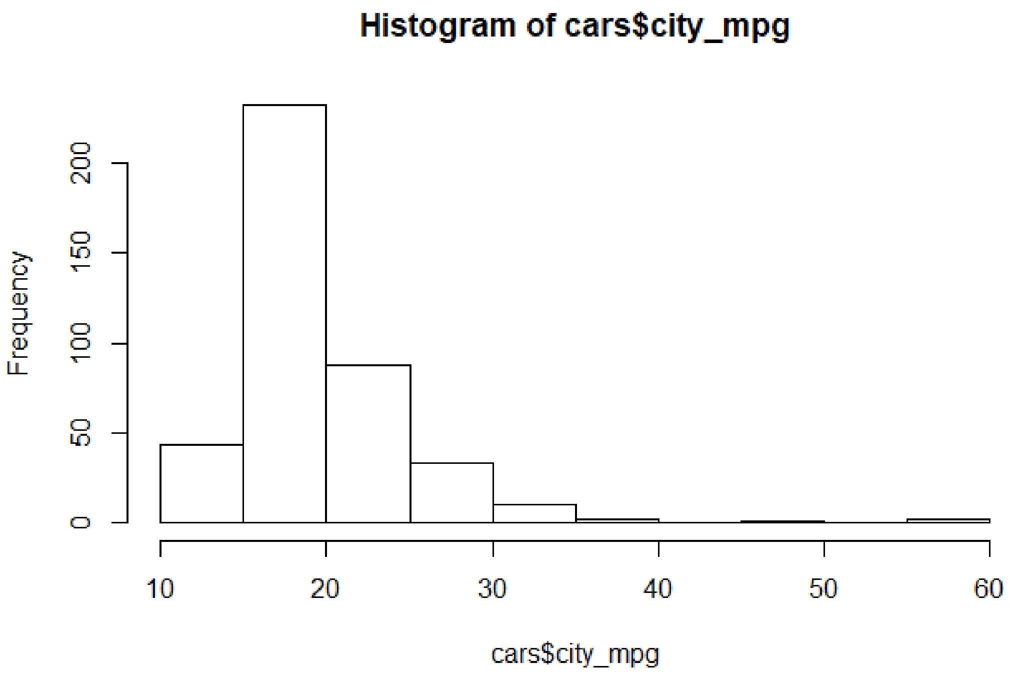
cars=read.csv('C:/Users/gowri/Desktop/Clg/R Programming/MS Lab/cars04.csv')
str(cars)

```
> cars=read.csv('C:/Users/gowri/Desktop/Clg/R Programming/MS Lab/cars04.csv')
> str(cars)
'data.frame':   428 obs. of  19 variables:
 $ name       : Factor w/ 425 levels "Acura 3.5 RL 4dr",..: 66 67 68 69 70 114 115 133 129 130 ...
 $ sports_car : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ suv        : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ wagon      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ minivan    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ pickup     : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ all_wheel  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ rear_wheel : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ msrp       : int  11690 12585 14610 14810 16385 13670 15040 13270 13730 15460 ...
 $ dealer_cost: int  10965 11802 13697 13884 15357 12849 14086 12482 12906 14496 ...
 $ eng_size   : num  1.6 1.6 2.2 2.2 2.2 2 2 2 2 2 ...
 $ ncyl       : int  4 4 4 4 4 4 4 4 4 4 ...
 $ horsepwr   : int  103 103 140 140 140 132 132 130 110 130 ...
 $ city_mpg   : int  28 28 26 26 26 29 29 26 27 26 ...
```

glimpse(cars)

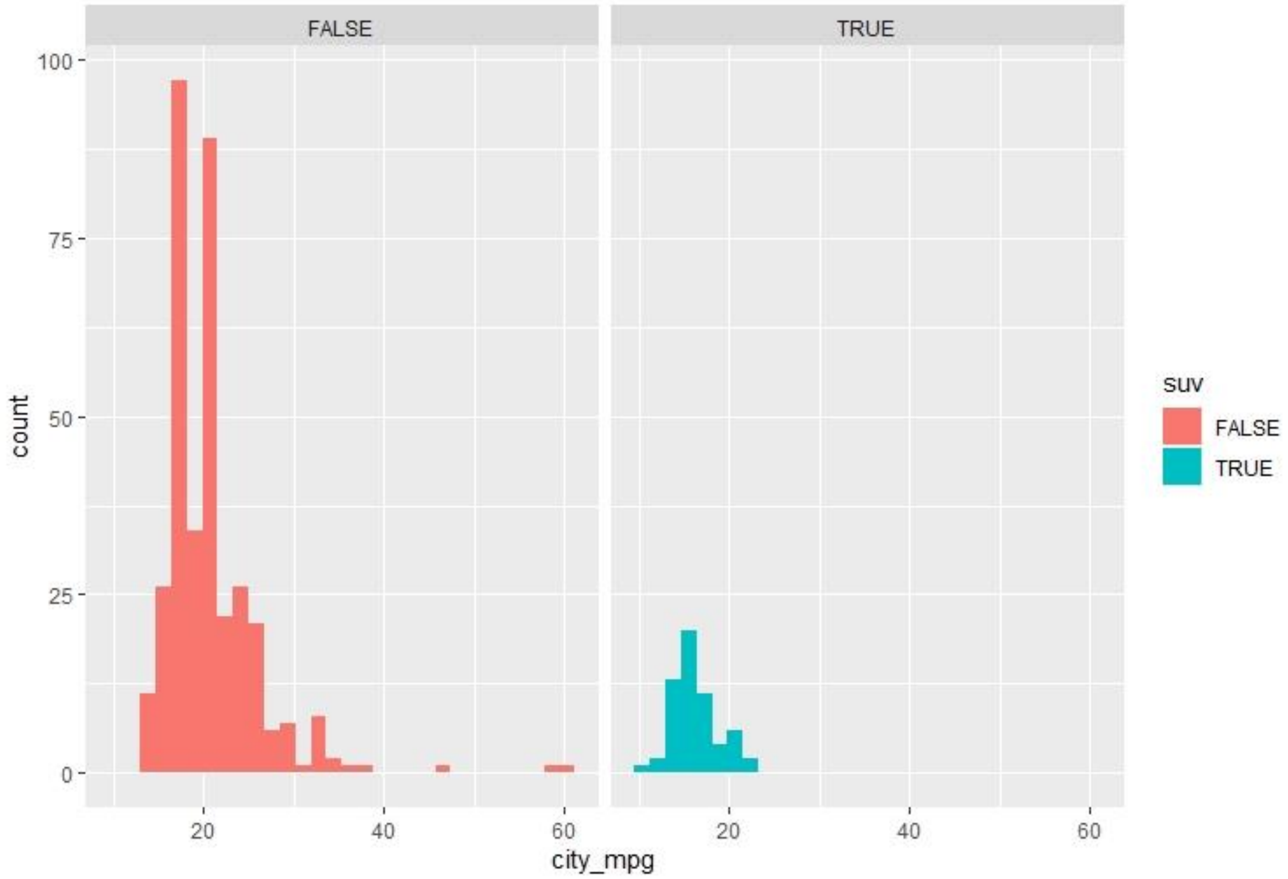
```
> glimpse(cars)
Observations: 428
Variables: 19
$ name        <fct> Chevrolet Aveo 4dr, Chevrolet Aveo LS 4dr hatch, Chevrolet Cavalier 2dr, Chevr...
$ sports_car  <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ suv         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ wagon       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ minivan     <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ pickup      <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ all_wheel   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ rear_wheel  <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
$ msrp        <int> 11690, 12585, 14610, 14810, 16385, 13670, 15040, 13270, 13730, 15460, 15580, 1...
$ dealer_cost <int> 10965, 11802, 13697, 13884, 15357, 12849, 14086, 12482, 12906, 14496, 14607, 1...
$ eng_size    <dbl> 1.6, 1.6, 2.2, 2.2, 2.2, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.7, 1.7, 1.7, 1.6, 1.6...
$ ncyl        <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ...
$ horsepwr    <int> 103, 103, 140, 140, 140, 132, 132, 130, 110, 130, 130, 115, 117, 115, 103, 103...
$ city_mpg    <int> 28, 28, 26, 26, 26, 29, 29, 26, 27, 26, 26, 32, 36, 32, 29, 29, 29, 26, 26, 26...
```

#Plot a hist of city Millage
hist(cars$city_mpg)

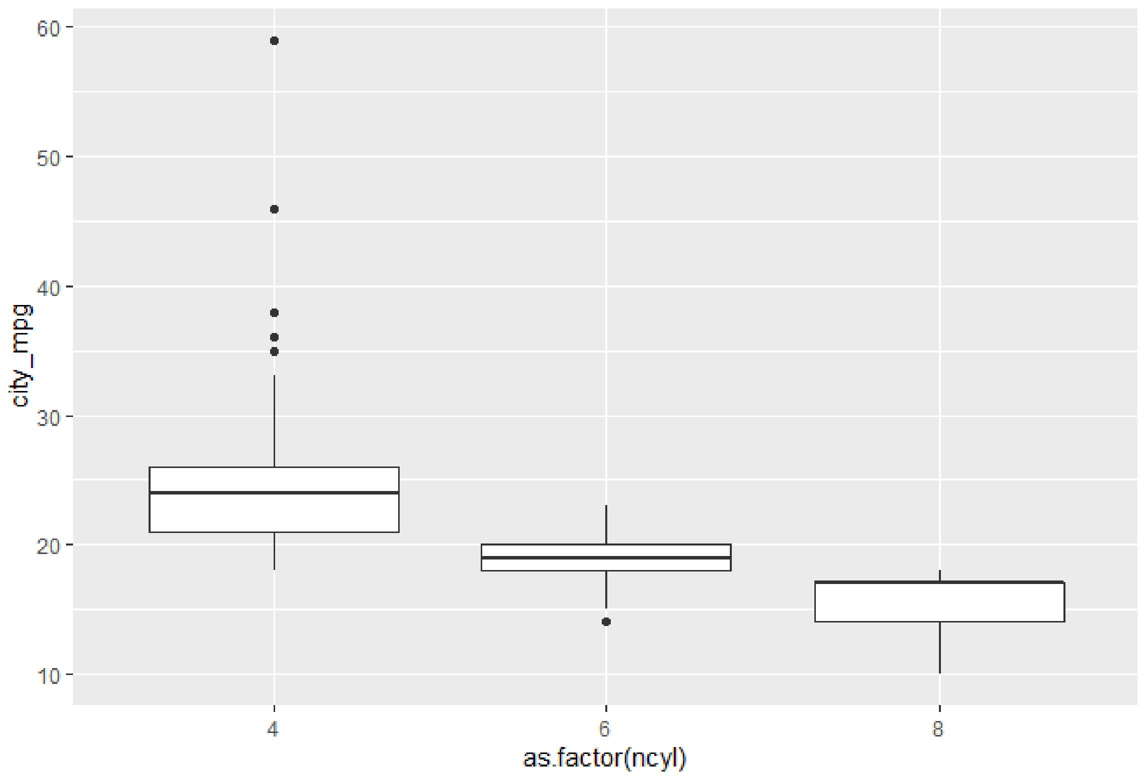


Roll No: 17E111

ggplot(cars,aes(x=city_mpg,fill=suv))+geom_histogram()+facet_wrap(~suv)



#BoxPlot and DensityPlot for 4,6 and 8 Cylinders
common_cyl=filter(cars,ncyl %in% c(4,6,8))
ggplot(common_cyl,aes(x=as.factor(ncyl),y=city_mpg))+geom_boxplot()



Result:

Thus, all the EDA operations for categorical and numerical data have been performed successfully.

Roll No: 17E111

Ex. No: 4
Date: 17-08-2019

Grammar of Graphics

Aim:

To study different charts in Grammar of Graphics by plotting different fancy plots.

Procedure:

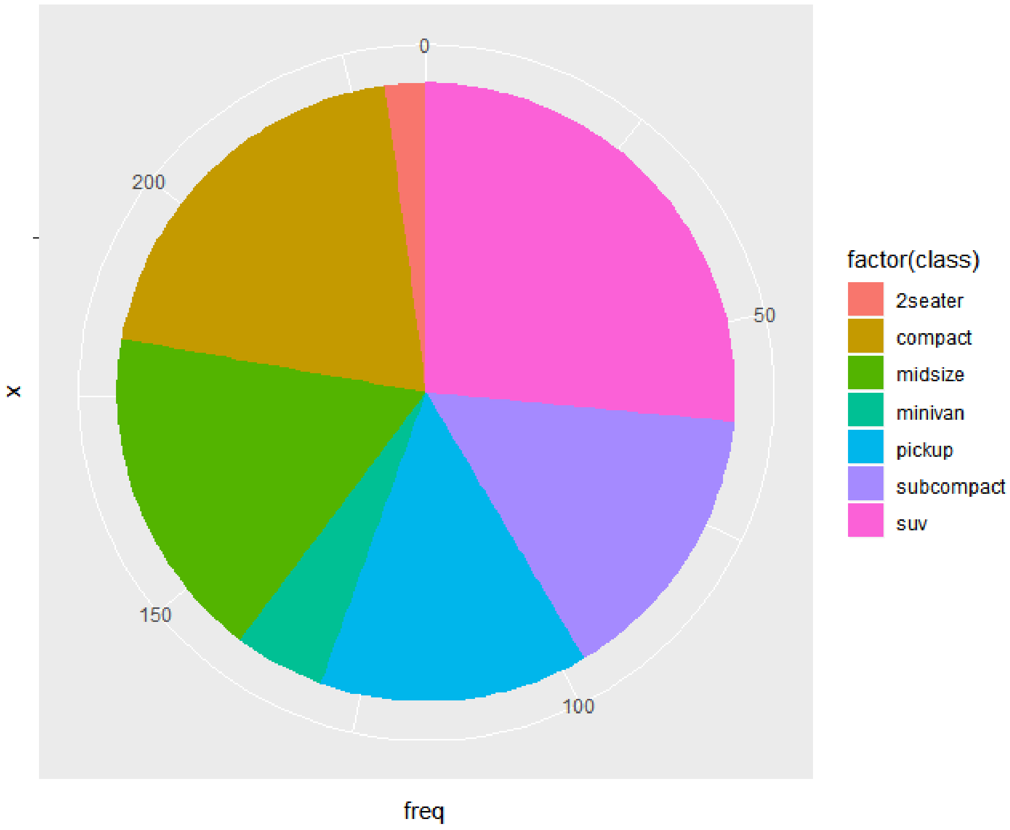
1. Open R studio.
2. In the script window perform all the **ggplot** visualization operations for categorical and numerical data.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

Program:

```
library(dplyr)
library(ggplot2)
library(tidyr)
comics=read.csv("comics.csv")
cars=read.csv("cars04.csv")
```

#Pie Chart

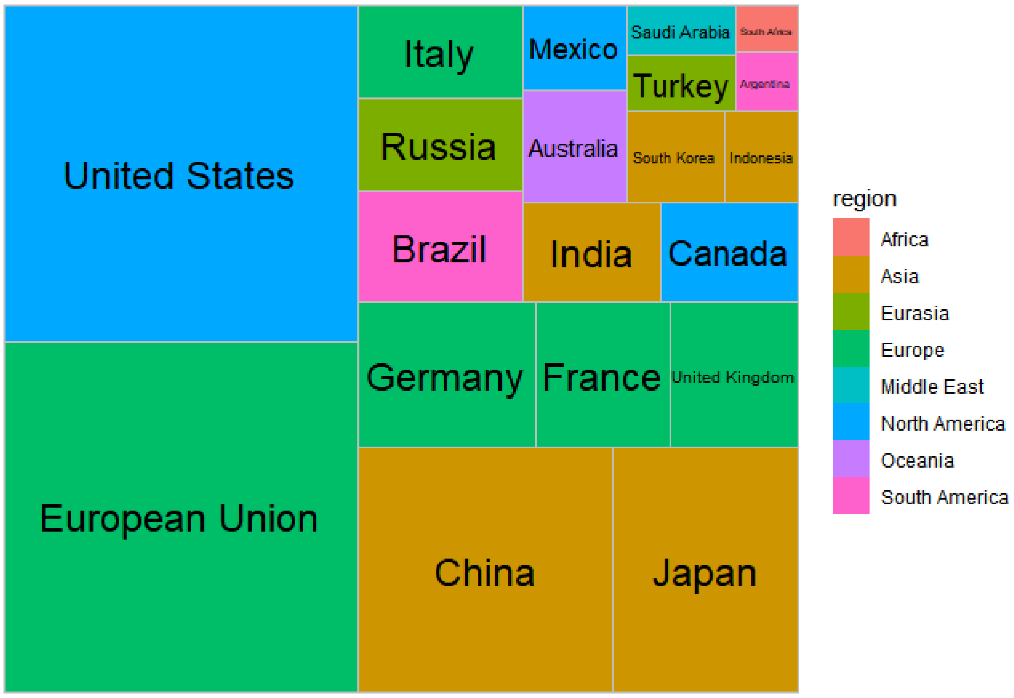
```
df=as.data.frame(table(mpg$class))
colnames(df) <- c("class", "freq")
ggplot(df, aes(x = "", y=freq, fill = factor(class))) +
  geom_bar( stat = "identity")+ coord_polar(theta = "y", start=0)
```



#Treemap Chart

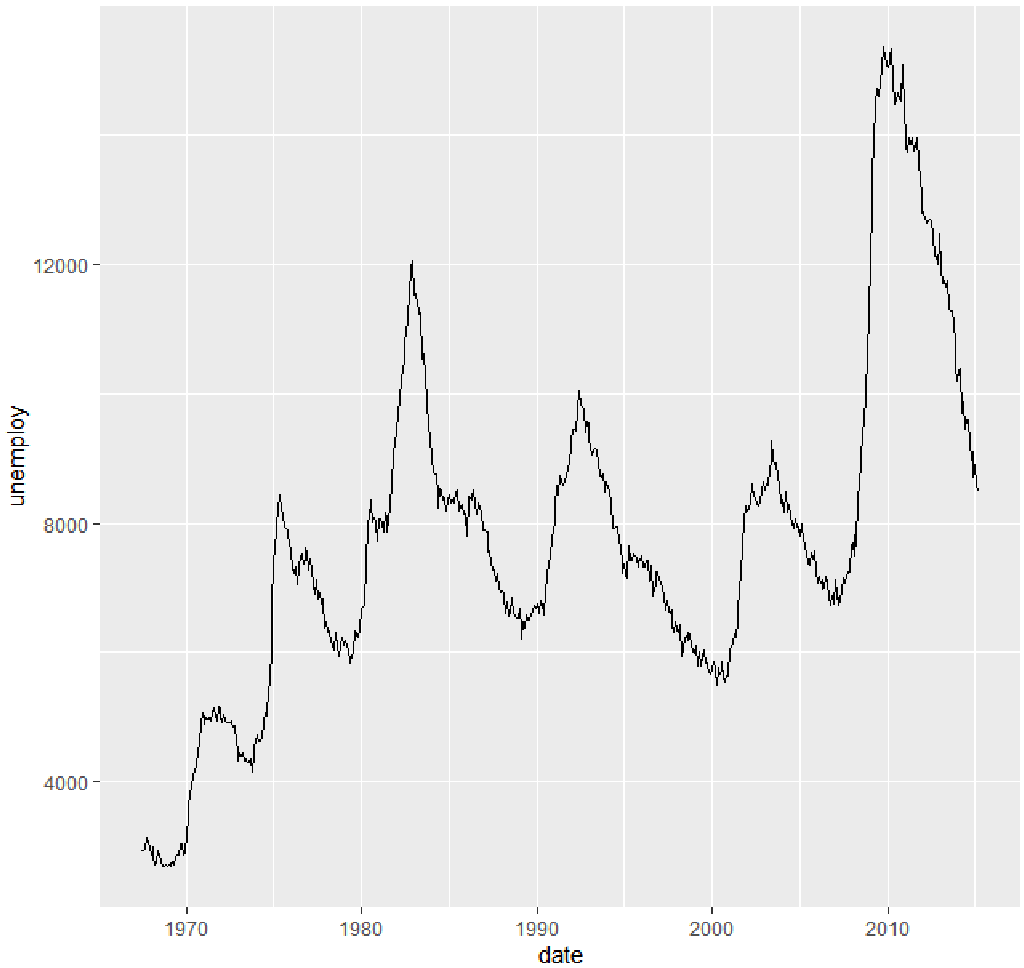
```
install.packages("treemapify")
library(treemapify)
library(dplyr)
library(ggplot2)
View(G20)
ggplot(G20, aes(area = gdp_mil_usd, fill = region, label = country)) +
  geom_treemap() +
  geom_treemap_text(  place = "centre")
```

Roll No: 17E111



#Time Series Chart

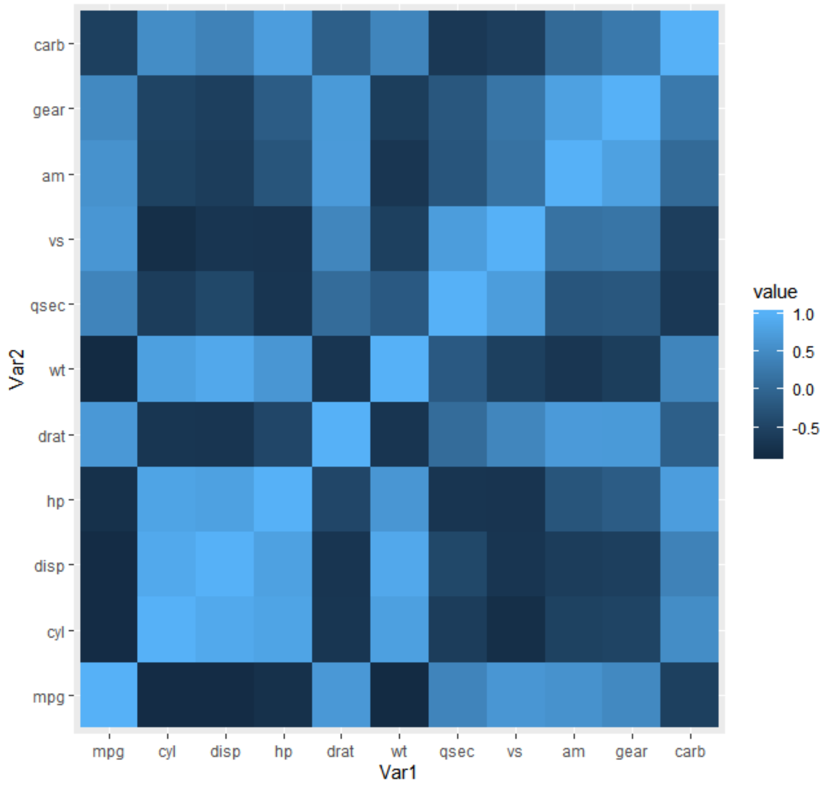
economics %>% ggplot(aes(x=date)) + geom_line(aes(y=unemploy))



Roll No: 17E111

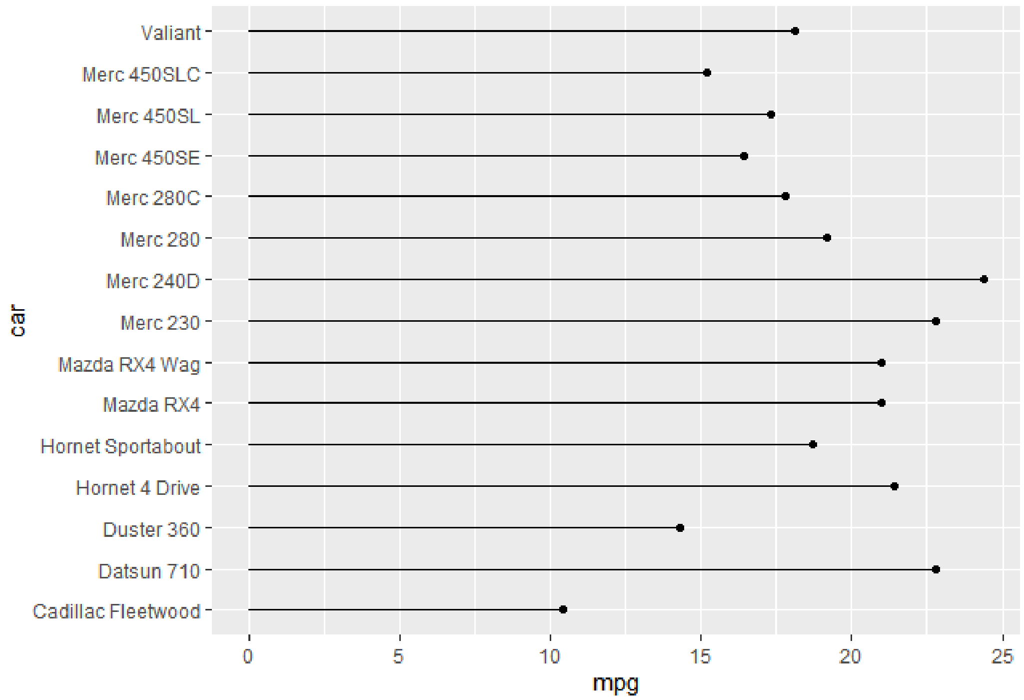
#Heat Map Chart

```
library(ggplot2)
library(reshape2)
mydata=mtcars
cormat = round(cor(mydata),2)
melted_cormat = melt(cormat)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) + geom_tile()
```



#Lollipop Plot Chart

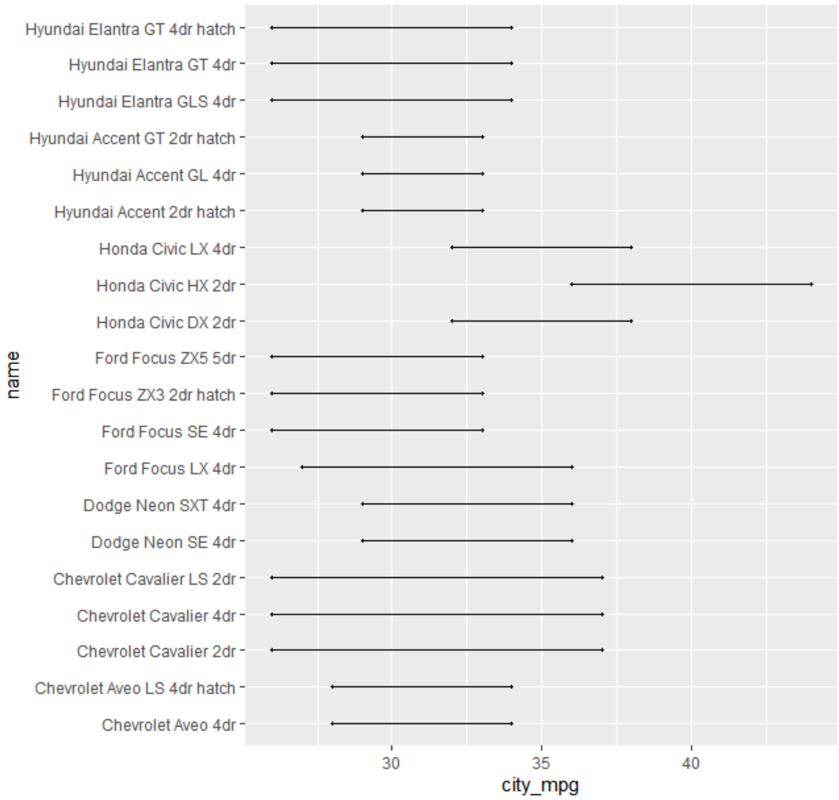
```
mtcars$car = row.names(mtcars)
head(mtcars , n=15) %>% ggplot( aes(x = mpg, y = car)) +
  geom_segment(aes(x = 0, y = car, xend = mpg, yend = car)) +
  geom_point()
```



Roll No: 17E111

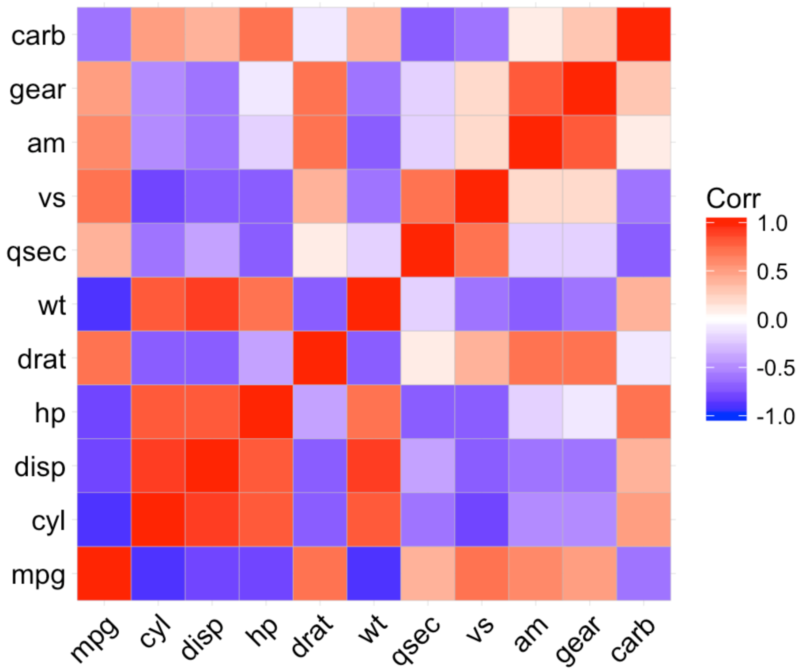
#Dumbbell Chart

```
library(ggplot2)
library(ggalt)
head(read.csv("C:/Users/gowri/Desktop/Clg/R Programming/MS Lab/cars04.csv"),20) %>%
  ggplot(aes(x=city_mpg, xend=hwy_mpg, y=name)) +
  geom_dumbbell()
```



#Correlogram Chart

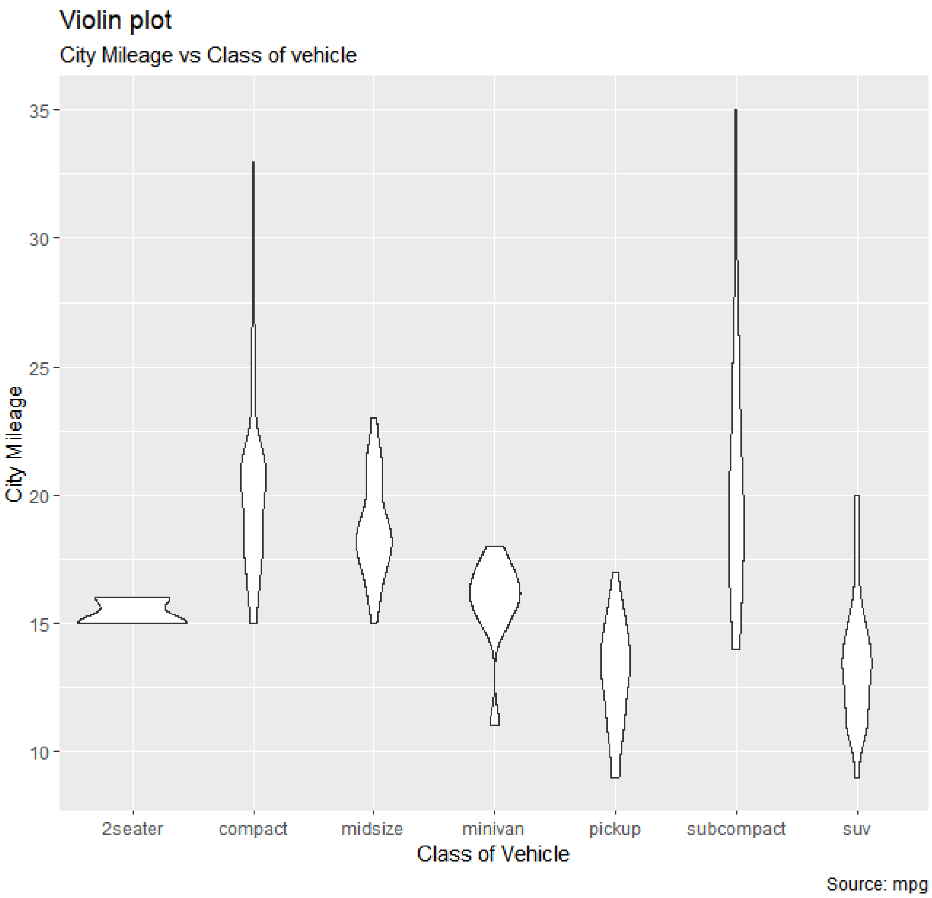
```
install.packages("ggcorrplot")
library(ggcorrplot)
data(mtcars)
corr <- round(cor(mtcars), 1)
head(corr[, 1:6])
p.mat <- cor_pmat(mtcars)
head(p.mat[, 1:4])
ggcorrplot(corr, hc.order = TRUE, outline.col = "white")
```



Roll No: 17E111

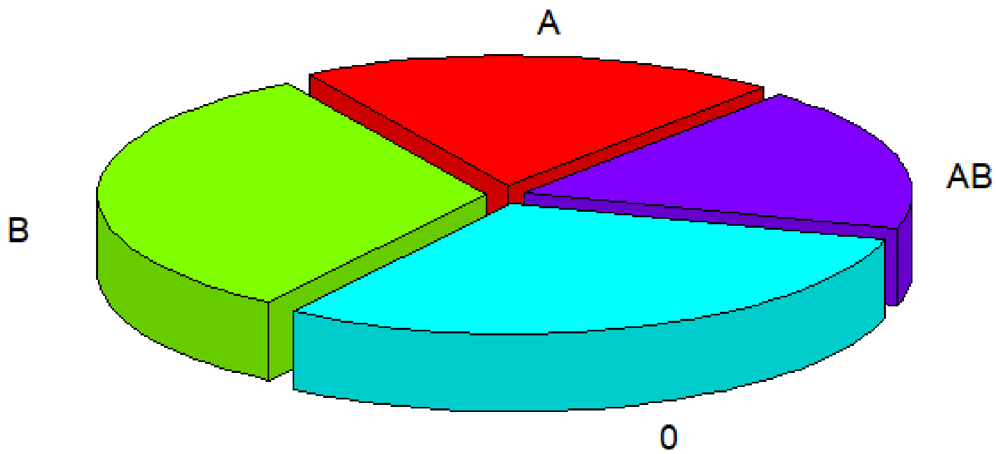
#Violin Plot

```
ggplot(mpg, aes(class, cty))+ geom_violin() +
  labs(title="Violin plot",
       subtitle="City Mileage vs Class of vehicle",
       caption="Source: mpg",
       x="Class of Vehicle",
       y="City Mileage")
```



#3D-Pie Chart

```
install.packages("plotrix")
library(plotrix)
mydata <- data.frame(group=c("A", "B", "0", "AB"), FR=c(20, 32, 30, 18))
pie3D(mydata$FR, labels = mydata$group, main = "An exploded 3D pie chart", explode=0.1, radius=.9, labelcex = 1.2,  start=0.7)
```



Result:

Thus, all the Grammar of Graphics Visualizations for different datasets have been performed successfully.

Roll No: 17E111

Ex. No: 5
Date: 19-08-2019

Sentimental Analysis in R

Aim:

To perform Sentimental Analysis using Tweeter dataset in R.

Procedure:

1. Open R studio.
2. In the script window load the datasets required and perform Sentimental Analysis.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

Program:

```
library(tidyr)
library(dplyr)
library(tidytext)

#1. geocoded_tweets has been pre-defined in share folder
load("geocoded_tweets.rda")
txt=read.delim("NRC_emotion_lexicon_list.txt",header = T)
colnames(txt)=c("word","sentiment","score")

#2. Use data frame with text data With inner join, implement sentiment analysis
# using `nrc` and assign to anobject tweet_nrc
txt=as.data.frame(txt,na.rm=T)
geocoded_tweets=as.data.frame(geocoded_tweets,na.rm=T)
tweets_nrc=geocoded_tweets %>% inner_join(txt)
View(tweets_nrc)
```

exp4.R × 19082019.R* × tweets_nrc ×

Filter

	state	word	freq	sentiment	score
1	alabama	abandoned	4070.773	anger	1
2	alabama	abandoned	4070.773	anticipation	0
3	alabama	abandoned	4070.773	disgust	0
4	alabama	abandoned	4070.773	fear	1
5	alabama	abandoned	4070.773	joy	0
6	alabama	abandoned	4070.773	negative	1
7	alabama	abandoned	4070.773	positive	0
8	alabama	abandoned	4070.773	sadness	1
9	alabama	abandoned	4070.773	surprise	0
10	alabama	abandoned	4070.773	trust	0
11	alabama	ability	12406.263	anger	0
12	alabama	ability	12406.263	anticipation	0
13	alabama	ability	12406.263	disgust	0
14	alabama	ability	12406.263	fear	0

Roll No: 17E111

#3. From the object tweet_nrc
# Filter to only choose the words associated with sadness
# Group by word
# Use the summarize verb to find the mean frequency
# Arrange to sort in order of descending frequency
sad_tweets=tweets_nrc %>% filter(sentiment=='sadness') %>% group_by(word)
View(sad_tweets)

	state	word	freq	sentiment	score
1	alabama	abandoned	4070.773	sadness	1
2	alabama	ability	12406.263	sadness	0
3	alabama	abortion	3267.394	sadness	1
4	alabama	absolute	22956.284	sadness	0
5	alabama	abuse	7185.962	sadness	1

sad_tweets=sad_tweets %>% summarize(freq=mean(freq))
View(sad_tweets)

	word	freq
1	abandon	4258.445
2	abandoned	6015.790
3	abandonment	6895.099
4	abba	6390.326
5	abbot	33404.724

sad_tweets=sad_tweets %>% arrange(desc(freq))
View(sad_tweets)

	word	freq
1	don	4211296.6
2	love	3878682.9
3	good	2826807.9
4	day	2817019.4
5	time	2475540.6

Roll No: 17E111

#4. From the object tweet_nrc do the following and assign to joy_words
Filter to choose only words associated with joy
Group by each word
Use the summarize verb to find the mean frequency
Arrange to sort in order of descending frequency

```
joy_words=tweets_nrc %>% filter(sentiment=="joy") %>% group_by(word) #%>%
        mutate(mean=mean(freq)) %>% ungroup()
View(joy_words)
```

exp4.R × | 19082019.R* × | joy_words ×

Filter

	state	word	freq	sentiment	score
1	alabama	abandoned	4070.773	joy	0
2	alabama	ability	12406.263	joy	0
3	alabama	abortion	3267.394	joy	0
4	alabama	absolute	22956.284	joy	0
5	alabama	abuse	7185.962	joy	0

```
joy_words=joy_words %>% summarize(freq=mean(freq))
joy_words=joy_words %>% arrange(desc(freq))
View(joy_words)
```

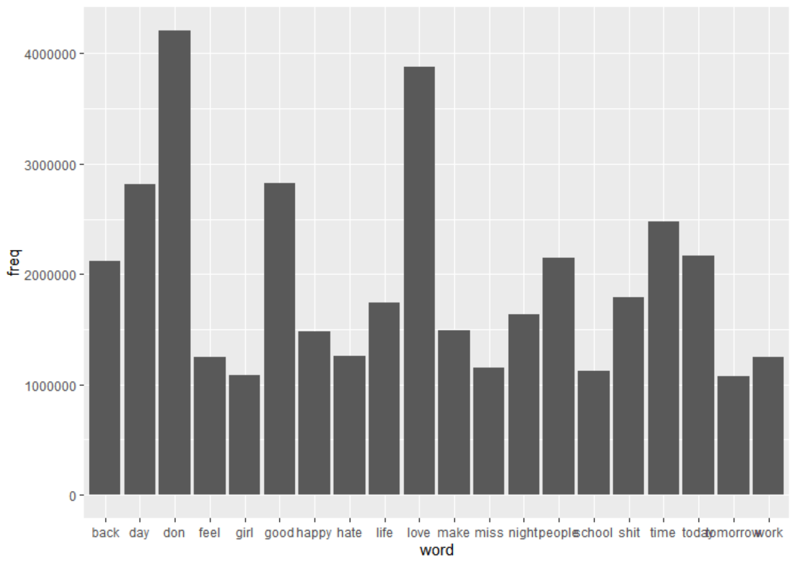
exp4.R × | 19082019.R* × | joy_words ×

Filter

	word	freq
1	don	4211296.6
2	love	3878682.9
3	good	2826807.9
4	day	2817019.4
5	time	2475540.6

#5. Load ggplot2 to extract top 20 joy words to plot bar plot having word in x and freq in y

```
library(ggplot2)
options(scipen=2)
joy_words %>% top_n(20) %>% ggplot(aes(x=word,y=freq))+geom_bar(stat='identity')
```



Roll No: 17E111

#6. Find only the words for the state of Utah and associated with joy
Arrange to sort in order of descending frequency
tweets_nrc %>% filter(state=='utah') %>% filter(sentiment=="joy") %>% arrange(desc(freq))

```
> # Arrange to sort in order of descending frequency
> tweets_nrc %>% filter(state=='utah') %>% filter(senti
   state       word      freq sentiment score
1   utah       love 4207322.0       joy     1
2   utah        don 3981874.0       joy     0
3   utah       good 3035113.5       joy     1
4   utah        day 2704126.6       joy     0
5   utah       time 2399778.9       joy     0
6   utah      today 2144214.7       joy     0
7   utah     people 2098067.3       joy     0
8   utah       back 1893829.7       joy     0
9   utah       life 1819456.8       joy     0
10  utah       park 1663328.2       joy     0
11  utah      night 1536655.4       joy     0
12  utah       make 1430144.2       joy     0
13  utah      happy 1402567.6       joy     1
```

# 7. Which states have the most positive Twitter users?
# assign to tweets_bing
tweets_bing=tweets_nrc %>% group_by(state) %>% filter(sentiment=='positive') %>% count(sentiment,sort=T)
tweets_bing

```
> tweets_bing
# A tibble: 49 x 3
# Groups:   state [49]
   state                sentiment     n
   <chr>                <fct>     <int>
 1 district of columbia positive   4863
 2 massachusetts        positive   4446
 3 vermont              positive   4400
 4 new hampshire        positive   4348
 5 maine                positive   4323
 6 colorado             positive   4290
 7 wisconsin            positive   4283
 8 new york             positive   4261
 9 washington           positive   4257
10 rhode island         positive   4228
# ... with 39 more rows
```

#8. From the tweets_bing object
# Group by two columns: state and sentiment
# Use summarize to calculate the mean frequency for these groups to spread(sentiment, freq)
# Calculate the ratio of positive to negative words
# Use aes() to put state on the x-axis and ratio on the y-axis
# Make a plot with points using geom_point()
tweets_bing=tweets_bing %>% group_by(state,sentiment)
colnames(tweets_bing)=c("state","sentiment","freq")
tweets_bing=tweets_bing %>% summarize(freq=mean(freq))
tweets_bing

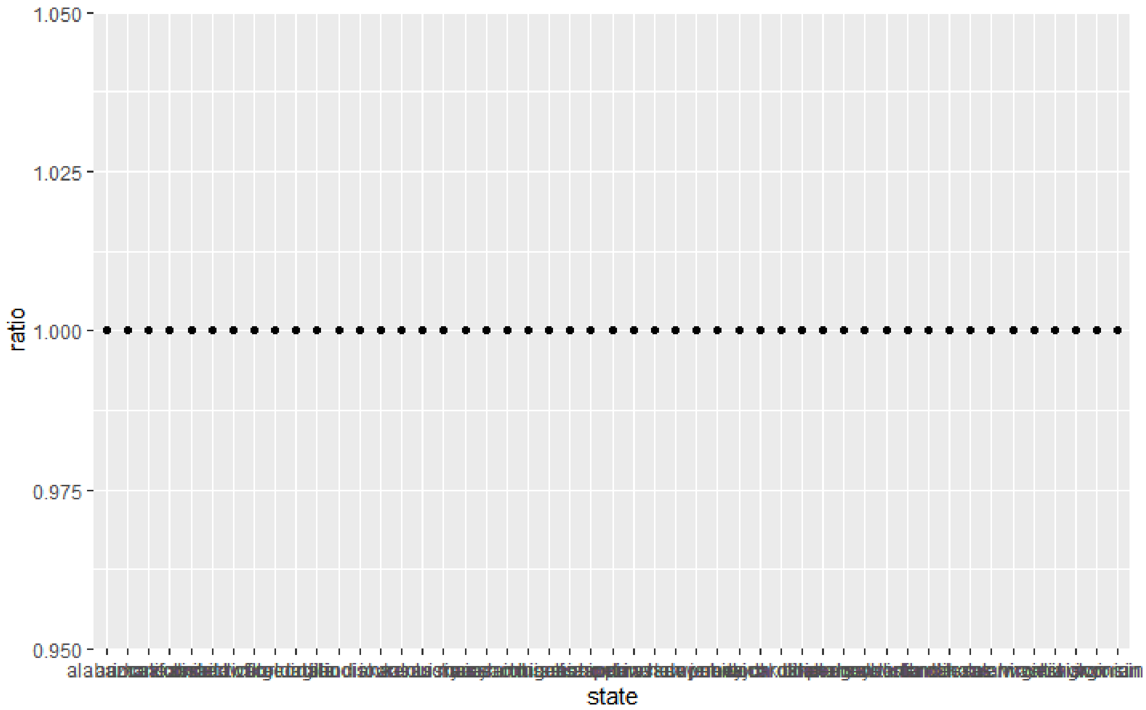
```
> tweets_bing=tweets_bing %>% summarize(freq=mean(freq))
> tweets_bing
# A tibble: 49 x 3
# Groups:   state [49]
   state                sentiment  freq
   <chr>                <fct>     <dbl>
 1 alabama              positive   3694
 2 arizona              positive   3910
 3 arkansas             positive   3879
 4 california           positive   4150
 5 colorado             positive   4290
 6 connecticut          positive   4208
 7 delaware             positive   3863
 8 district of columbia positive   4863
 9 florida              positive   4030
10 georgia              positive   3756
# ... with 39 more rows
```

Roll No: 17E111

temp=tweets_nrc %>% filter() %>%spread(sentiment, freq)
temp

```
> temp=tweets_nrc %>% filter() %>%spread(sentiment, freq)
> temp
     state       word score     anger anticipation   disgust      fear       joy  negative
1  alabama  abandoned     0        NA     4070.773  4070.773        NA  4070.773        NA
2  alabama  abandoned     1  4070.773           NA        NA  4070.773        NA  4070.773
3  alabama    ability     0 12406.263    12406.263 12406.263 12406.263 12406.263 12406.263
4  alabama    ability     1        NA           NA        NA        NA        NA        NA
5  alabama   abortion     0  3267.394     3267.394        NA        NA  3267.394        NA
6  alabama   abortion     1        NA           NA  3267.394  3267.394        NA  3267.394
7  alabama   absolute     0 22956.284    22956.284 22956.284 22956.284 22956.284 22956.284
8  alabama   absolute     1        NA           NA        NA        NA        NA        NA
9  alabama      abuse     0        NA     7185.962        NA        NA  7185.962        NA
10 alabama      abuse     1  7185.962           NA  7185.962  7185.962        NA  7185.962
11 alabama    academy     0 12483.919    12483.919 12483.919 12483.919 12483.919 12483.919
12 alabama    academy     1        NA           NA        NA        NA        NA        NA
13 alabama     accent     0 13445.534    13445.534 13445.534 13445.534 13445.534 13445.534
14 alabama     accept     0 42042.580    42042.580 42042.580 42042.580 42042.580 42042.580
```

ratio=sum(temp$positive,na.rm=T)/sum(temp$negative,na.rm=T)
tweets_bing %>% ggplot(aes(x=state,y=ratio))+geom_point()



Roll No: 17E111

```
#9. Install the packages tm, wordcloud, RColorBrewer
#From joy_words of word display the chart wordcloud
install.packages("tm")
install.packages("wordcloud")
install.packages("RColorBrewer")

library("tm")
library("wordcloud")
library("RColorBrewer")

wordcloud(head(joy_words$word,200),head(joy_words$freq,n=200))
```

Output:

Result:

Thus, Sentimental Analysis of the given datasets have been done successfully and a word-cloud has been designed for the positive words.

Roll No: 17E111

Ex. No: 6
Date: 26-08-2019

Simple Linear Regression in R

Aim:

To perform the Simple Linear Regression (SLR) of the given data in R.

Procedure:

1. Open R studio.
2. In the script window load the datasets required and perform Simple Linear Regression.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

Program:

```
#Import the data and assign to a Object
library(caTools)
ds=read.csv("https://raw.githubusercontent.com/guru99-edu/R-Programming/master/women.csv")
ds
```

```
> ds
    X height weight
1   1     58    115
2   2     59    117
3   3     60    120
4   4     61    123
5   5     62    126
6   6     63    129
7   7     64    132
8   8     65    135
9   9     66    139
10 10     67    142
11 11     68    146
```

```
#Split the dataset to training_set and testing_set
set.seed(123)
split =sample.split(ds$weight,SplitRatio=2/3)
training_set=subset(ds,split==TRUE)
testing_set = subset(ds,split==FALSE)
training_set
```

```
> training_set
    X height weight
1   1     58    115
3   3     60    120
6   6     63    129
7   7     64    132
9   9     66    139
10 10     67    142
12 12     69    150
13 13     70    154
14 14     71    159
15 15     72    164
```

```
testing_set
```

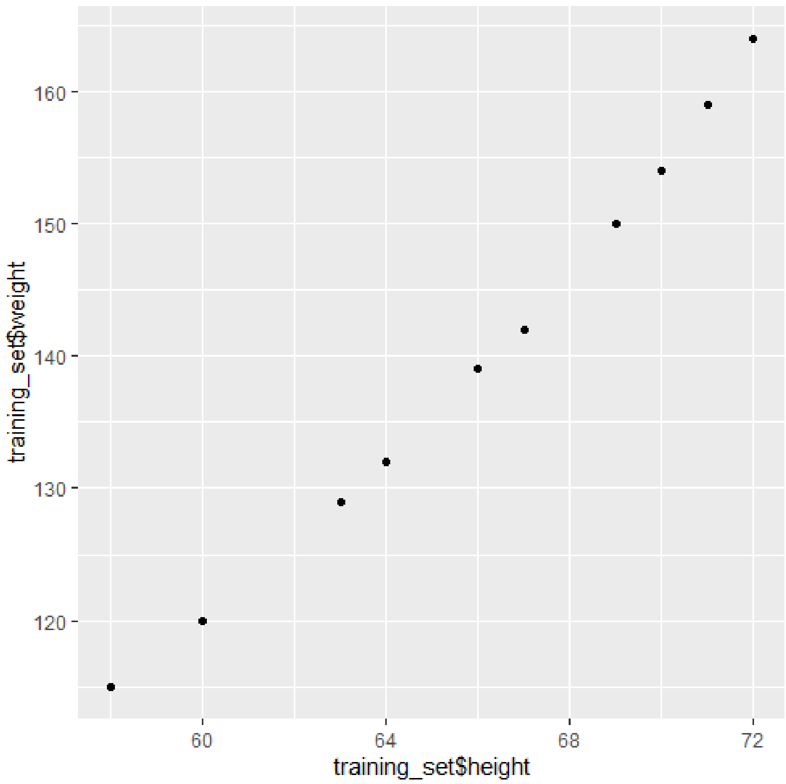
```
> testing_set
    X height weight
2   2     59    117
4   4     61    123
5   5     62    126
8   8     65    135
11 11     68    146
```

Roll No: 17E111

```
#Fitting the Regressor
regressor=lm(formula=weight~height,data=training_set)
summary(regressor)
```

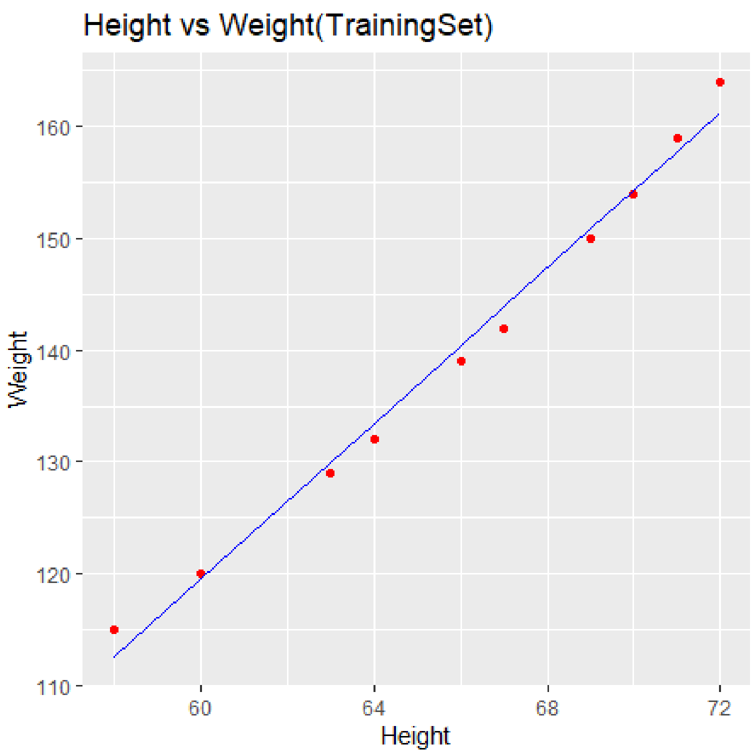
```
> summary(regressor)

Call:
lm(formula = weight ~ height, data = training_set)

Residuals:
   Min     1Q Median     3Q    Max 
 -1.88  -1.29  -0.58   1.02   2.72 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -89.2800     8.1557  -10.95 4.30e-06 ***
height        3.4800     0.1233   28.23 2.68e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.744 on 8 degrees of freedom
Multiple R-squared:  0.9901,	Adjusted R-squared:  0.9888 
F-statistic: 796.7 on 1 and 8 DF,  p-value: 2.681e-09
```

```
y_pred<-predict(regressor,testing_set)
#Visualize the SLR using ggplot2
library(ggplot2)
ggplot()+geom_point(aes(x=training_set$height,y=training_set$weight))
```




```
#Training_set
ggplot()+geom_point(aes(x=training_set$height,y=training_set$weight),color="red")+
  geom_line(aes(x=training_set$height,y=predict(regressor,training_set)),color="blue")+
  ggtitle("Height vs Weight(TrainingSet)")+xlab("Height")+ylab("Weight")
```

Roll No: 17E111




```
#Testing_set
ggplot()+geom_point(aes(x=testing_set$height,y=testing_set$weight),color="red")+
  geom_line(aes(x=testing_set$height,y=predict(regressor,testing_set)),color="blue")+
  ggtitle("Height vs Weight(TestSet)")+xlab("Height")+ylab("Weight")
```

Result:

Thus, the Simple Linear Regression (SLR) of R has been completed successfully.

Roll No: 17E111

Ex. No: 7
Date: 12-09-2019

Polynomial Regression in R

Aim:

To perform the Polynomial Regression of the given data in R.

Procedure:

1. Open R studio.
2. In the script window load the datasets required and perform Polynomial Regression.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

Program:

```
#Polynomial Regression
#
ds=read.csv("Position_Salaries.csv")
ds
```

```
> ds
            Position Level  Salary
1   Business Analyst     1   45000
2  Junior Consultant     2   50000
3  Senior Consultant     3   60000
4            Manager     4   80000
5    Country Manager     5  110000
6     Region Manager     6  150000
7            Partner     7  200000
8     Senior Partner     8  300000
9            C-level     9  500000
10               CEO    10 1000000
```

```
ds=ds[2:3]
#Splitting Dataset - Not Required since dataset has only 10 obs
options(scipen = 2)
#Feature Scaling - Not Required
#plot(ds$Level,ds$Salary)

#Fitting Linear Regression to the Dataset
poly_regressor=lm(formula = Salary ~ .,data = ds)
summary(poly_regressor)
```

```
> summary(poly_regressor)

Call:
lm(formula = Salary ~ ., data = ds)

Residuals:
    Min      1Q  Median      3Q     Max 
-170818 -129720  -40379   65856  386545 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  -195333     124790  -1.565  0.15615   
Level          80879      20112   4.021  0.00383 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 182700 on 8 degrees of freedom
Multiple R-squared:  0.669,	Adjusted R-squared:  0.6277 
F-statistic: 16.17 on 1 and 8 DF,  p-value: 0.003833
```

Roll No: 17E111

#Visualizing Linear Regression
library(ggplot2)
ggplot()+geom_point(aes(x=ds$Level,y=ds$Salary,color="red"))+
 geom_line(aes(x=ds$Level,y=predict(poly_regressor,newdata = ds)),colour="blue")

#Fitting Polynomial Regression x**2 to the Dataset
ds$Level2=ds$Level**2
ds

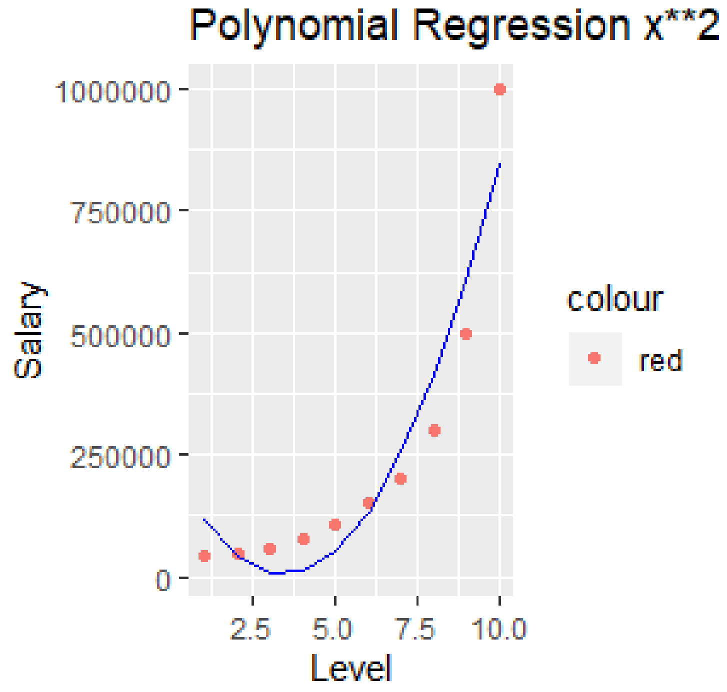
```
> ds$Level2=ds$Level**2
> ds
   Level  Salary Level2
1      1   45000      1
2      2   50000      4
3      3   60000      9
4      4   80000     16
5      5  110000     25
6      6  150000     36
7      7  200000     49
8      8  300000     64
9      9  500000     81
10    10 1000000    100
```

poly_regressor=lm(formula = Salary ~ .,data = ds)
summary(poly_regressor)

```
> summary(poly_regressor)

Call:
lm(formula = Salary ~ ., data = ds)

Residuals:
    Min      1Q  Median      3Q     Max 
-112833  -68852   10682   55186  153364 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)   232167     115571   2.009  0.08451 . 
Level        -132871      48268  -2.753  0.02839 * 
Level2         19432       4276   4.544  0.00265 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 98260 on 7 degrees of freedom
Multiple R-squared:  0.9162,	Adjusted R-squared:  0.8923 
F-statistic: 38.27 on 2 and 7 DF,  p-value: 0.0001703
```

Roll No: 17E111

```
#Visualizing Polynomial x**2 Regression
ggplot()+geom_point(aes(x=ds$Level,y=ds$Salary,color="red"))+
  geom_line(aes(x=ds$Level,y=predict(poly_regressor,newdata = ds)),colour="blue")+
  labs(title="Polynomial Regression x**2",x="Level",y="Salary")
```

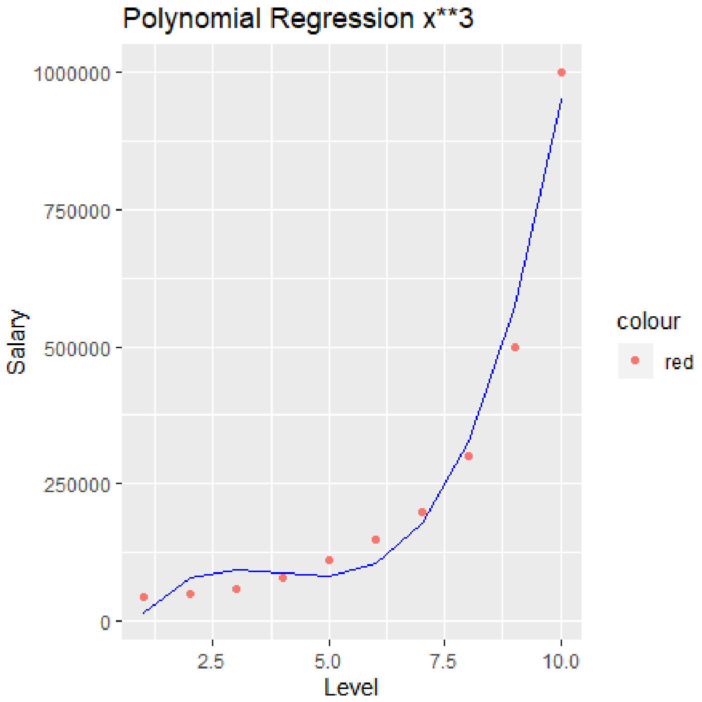



```
#Fitting Polynomial Regression x**3 to the Dataset
ds$Level3=ds$Level**3
poly_reg=lm(formula = Salary ~ .,data = ds)
summary(poly_reg)
```

```
> summary(poly_reg)

Call:
lm(formula = Salary ~ ., data = ds)

Residuals:
   Min     1Q Median     3Q    Max 
-75695 -28148   7091  29256  49538 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) -121333.3    97544.8  -1.244  0.25994   
Level        180664.3    73114.5   2.471  0.04839 * 
Level2       -48549.0    15081.0  -3.219  0.01816 * 
Level3         4120.0      904.3   4.556  0.00387 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 50260 on 6 degrees of freedom
Multiple R-squared:  0.9812,	Adjusted R-squared:  0.9718 
F-statistic: 104.4 on 3 and 6 DF,  p-value: 0.00001441
```

```
#Visualizing Polynomial x**3 Regression
ggplot()+geom_point(aes(x=ds$Level,y=ds$Salary,color="red"))+
  geom_line(aes(x=ds$Level,y=predict(poly_reg,newdata = ds)),colour="blue")+
  labs(title="Polynomial Regression x**3",x="Level",y="Salary")
```

Roll No: 17E111



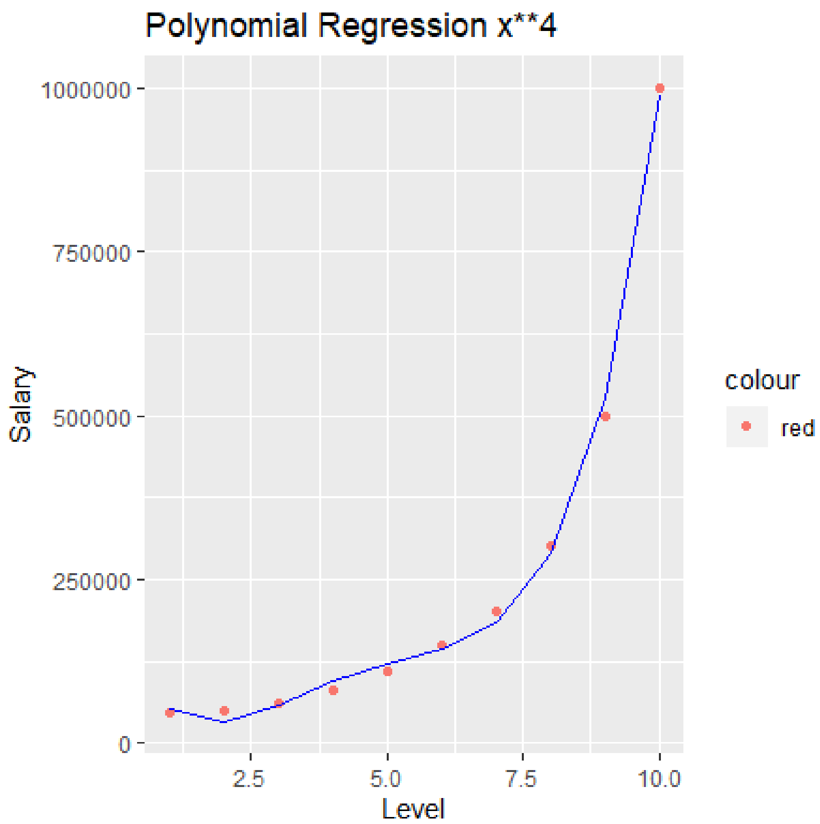

```
#Fitting Polynomial Regression x**4 to the Dataset
ds$Level4=ds$Level**4
poly_regr=lm(formula = Salary ~ .,data = ds)
summary(poly_regr)
```

```
> summary(poly_regr)

Call:
lm(formula = Salary ~ ., data = ds)

Residuals:
     1      2      3      4      5      6      7      8      9     10 
 -8357  18240   1358 -14633 -11725   6725  15997  10006 -28695  11084 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept) 184166.7    67768.0   2.718  0.04189 * 
Level      -211002.3    76382.2  -2.762  0.03972 * 
Level2       94765.4    26454.2   3.582  0.01584 * 
Level3      -15463.3     3535.0  -4.374  0.00719 **
Level4         890.2      159.8   5.570  0.00257 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20510 on 5 degrees of freedom
Multiple R-squared:  0.9974,	Adjusted R-squared:  0.9953 
F-statistic: 478.1 on 4 and 5 DF,  p-value: 0.000001213
```

#Fitting Polynomial Regression x**4 to the Dataset
ds$Level4=ds$Level**4
poly_regr=lm(formula = Salary ~ .,data = ds)
summary(poly_regr)

```
> summary(poly_regr)

Call:
lm(formula = Salary ~ ., data = ds)

Residuals:
     1      2      3      4      5      6      7      8      9     10 
 -8357  18240   1358 -14633 -11725   6725  15997  10006 -28695  11084 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept)  184166.7    67768.0   2.718  0.04189 * 
Level       -211002.3    76382.2  -2.762  0.03972 * 
Level2        94765.4    26454.2   3.582  0.01584 * 
Level3       -15463.3     3535.0  -4.374  0.00719 **
Level4          890.2      159.8   5.570  0.00257 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20510 on 5 degrees of freedom
Multiple R-squared:  0.9974,	Adjusted R-squared:  0.9953 
F-statistic: 478.1 on 4 and 5 DF,  p-value: 0.000001213
```

```
#Visualizing Polynomial x**4 Regression
ggplot()+geom_point(aes(x=ds$Level,y=ds$Salary,color="red"))+
  geom_line(aes(x=ds$Level,y=predict(poly_regr,newdata = ds)),colour="blue")+
  labs(title="Polynomial Regression x**4",x="Level",y="Salary")
```



Result:

Thus, the Polynomial Regression of R has been completed successfully.

Roll No: 17E111

Ex. No: 8
Date: 16-09-2019

Logistic Regression in R

Aim:

To perform the Logistic Regression of the given data in R.

Procedure:

1. Open R studio.
2. In the script window load the datasets required and perform Logistic Regression.
3. After performing all the operations, Click on Save button to save the document in local disk.
4. Exit from R studio.

Program:

```
#Logistic Regression
#Importing and Pre-processing the dataset
ds=read.csv("Social_Network_Ads.csv")
ds
```

```
> ds
   User.ID Gender Age EstimatedSalary Purchased
1 15624510   Male  19           19000         0
2 15810944   Male  35           20000         0
3 15668575 Female  26           43000         0
4 15603246 Female  27           57000         0
5 15804002   Male  19           76000         0
6 15728773   Male  27           58000         0
7 15598044 Female  27           84000         0
8 15694829 Female  32          150000         1
9 15600575   Male  25           33000         0
```

```
ds=ds[-1:-2]
ds
```

```
> ds
    Age EstimatedSalary Purchased
1    19           19000         0
2    35           20000         0
3    26           43000         0
4    27           57000         0
5    19           76000         0
6    27           58000         0
7    27           84000         0
8    32          150000         1
9    25           33000         0
10   35           65000         0
```

```
#Taking Care of Missing Data
summary(ds) #No missing data found
```

```
> summary(ds) #No missing data found
      Age        EstimatedSalary     Purchased
 Min.   :18.00   Min.   : 15000   Min.   :0.0000
 1st Qu.:29.75   1st Qu.: 43000   1st Qu.:0.0000
 Median :37.00   Median : 70000   Median :0.0000
 Mean   :37.66   Mean   : 69743   Mean   :0.3575
 3rd Qu.:46.00   3rd Qu.: 88000   3rd Qu.:1.0000
 Max.   :60.00   Max.   :150000   Max.   :1.0000
```

```
#Encoding Categorial Data
```

Roll No: 17E111

class(ds$Purchased) #Integer
ds$Purchased = as.factor(ds$Purchased)
ds

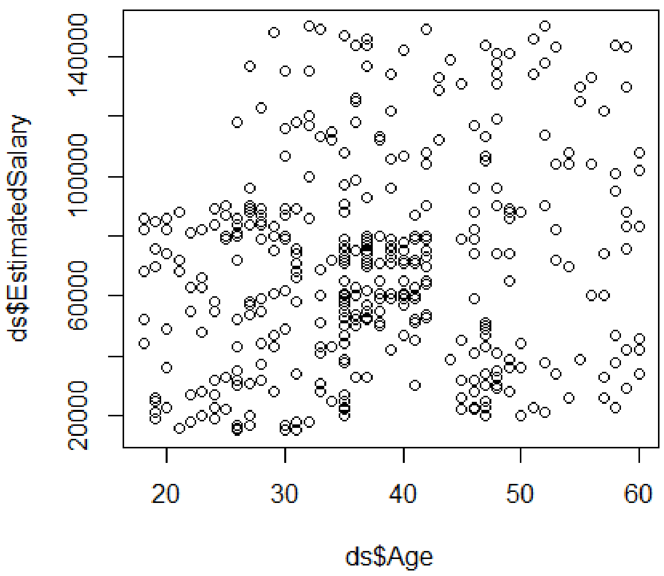
```
> #Encoding Categorial Data
> class(ds$Purchased) #Integer
[1] "integer"
> ds$Purchased = as.factor(ds$Purchased)
> #Encoding Categorial Data
> class(ds$Purchased) #Integer
[1] "factor"
```

#Splitting the dataset into training and testing set
library(caTools)
set.seed(123)
split=sample.split(ds$Purchased,SplitRatio = 0.75)
split

```
> split=sample.split(ds$Purchased,SplitRatio = 0.75)
> split
  [1]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
 [21]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [41]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
 [81]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
[101]  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE
[121]  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
[141]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

training_set = subset(ds,split==T)
testing_set = subset(ds,split==F)

#Feature Scaling
plot(dsAge,dsEstimatedSalary)



training_set[,-3] = scale(training_set[,-3])
testing_set[,1:2] = scale(testing_set[,1:2])

Roll No: 17E111

training_set

```
> training_set
          Age EstimatedSalary Purchased
1  -1.76554750     -1.47334137         0
3  -1.09629664     -0.78837605         0
6  -1.00068938     -0.36027273         0
7  -1.00068938      0.38177303         0
8  -0.52265305      2.26542765         1
10 -0.23583125     -0.16049118         0
11 -1.09629664      0.26761214         0
13 -1.66994024      0.43885347         0
14 -0.52265305     -1.50188159         0
```

testing_set

```
> training_set
          Age EstimatedSalary Purchased
1  -1.76554750     -1.47334137         0
3  -1.09629664     -0.78837605         0
6  -1.00068938     -0.36027273         0
7  -1.00068938      0.38177303         0
8  -0.52265305      2.26542765         1
10 -0.23583125     -0.16049118         0
11 -1.09629664      0.26761214         0
13 -1.66994024      0.43885347         0
14 -0.52265305     -1.50188159         0
15 -1.86115477      0.32469259         0
```

```
#Fitting Logistic Regression to training_set
classifier = glm(formula = Purchased ~ .,family = quasibinomial(),data = training_set)
summary(classifier)
```

```
> summary(classifier)

Call:
glm(formula = Purchased ~ ., family = quasibinomial(), data = training_set)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-3.0753  -0.5235  -0.1161   0.3224   2.3977  

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)      -1.1923     0.2216  -5.380 1.51e-07 ***
Age               2.6324     0.3800   6.926 2.68e-11 ***
EstimatedSalary   1.3947     0.2554   5.460 1.01e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 1.205802)

    Null deviance: 390.89  on 299  degrees of freedom
Residual deviance: 199.78  on 297  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

```
prob_predict = predict.glm(classifier,testing_set[-3],type = "response")
y_pred = round(prob_predict)
```

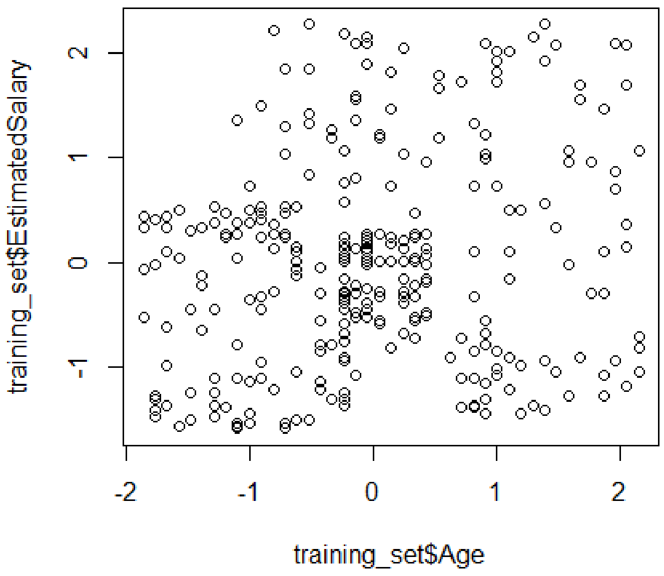
Roll No: 17E111

#Confusion Matrix to check the accuracy of the prediction
table(testing_set[,3],y_pred>0.5)

```
> table(testing_set[,3],y_pred>0.5)

    FALSE TRUE
  0    57    7
  1    10   26
```

#Visualizing the training_set result
install.packages("ElemStatLearn")
library(ElemStatLearn)

plot(training_set$Age, training_set$EstimatedSalary)



set=training_set
X1=seq(min(set[,1])-1, max(set[,1])+1,by=0.01)
X2=seq(min(set[,2])-1, max(set[,2])+1,by=0.01)
grid_set = expand.grid(X1,X2)
colnames(grid_set)=c('Age', 'EstimatedSalary')
prob_set = predict(classifier, type='response', newdata=grid_set)
y_grid=ifelse(prob_set>0.5,1,0)
plot(set[,-3], main="Logistic Regression - Training",
    xlab = "Age", ylab="Estimated Salary",xlim=range(X1), ylim=range(X2))

```
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add=T)
```


```
points(grid_set, pch='.', col=ifelse(y_grid==1, 'springgreen3', 'tomato'))
points(set, pch=21, bg=ifelse(set[,3]==1,'green4', 'red3'))
```

Result:

Thus, the Logistic Regression of R has been completed successfully.

Roll No: 17E111

Ex. No: 9

Date: 21-09-2019

Decision Tree in R

Aim:

To analyse the survival rate by creating a Decision Tree of given dataset (Titanic) in R.

Procedure:

1. Open R studio.
2. In the script window, load the dataset required and build a Decision Tree Model.
3. After performing all the operations, Click on save button to save the document in local disk.
4. Exit from R studio.

Program:

```
#Decision Tree = Titanic Dataset
#1. Import Dataset
titanic=read.csv("https://raw.githubusercontent.com/guru99-edu/R-Programming/master/titanic_data.csv")
View(titanic)
```

24092019.R × 19092019.R × titanic ×

Filter

	x	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	home.dest
1	1	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	St Louis, MO
2	2	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26	S	Montreal, PQ / Che:
3	3	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26	S	Montreal, PQ / Che:
4	4	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26	S	Montreal, PQ / Che:
5	5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26	S	Montreal, PQ / Che:
6	6	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.55	E12	S	New York, NY
7	7	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S	Hudson, NY
8	8	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36	S	Belfast, NI
9	9	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S	Bayside, Queens, N'
10	10	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	?	C	Montevideo, Urugu

```
set.seed(123)
#2. Shuffling the Index
shuffel_index=sample(1:nrow(titanic))
titan = titanic[shuffel_index,]
View(titan)
```

24092019.R × 19092019.R* × titan ×

Filter

	x	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	hom
377	377	2	1	Collett, Mr. Sidney C Stuart	male	24	0	0	28034	10.5	?	S	Lonc
1032	1032	3	0	Morley, Mr. William	male	34	0	0	364506	8.05	?	S	?
535	535	2	1	Phillips, Miss. Kate Florence ('Mrs Kate Louise Phillips Marsh...	female	19	0	0	250655	26	?	S	Wor
1154	1154	3	0	Robins, Mrs. Alexander A (Grace Charity Laury)	female	47	1	0	A/5. 3337	14.5	?	S	?
1228	1228	3	0	Strandberg, Miss. Ida Sofia	female	22	0	0	7553	9.8375	?	S	?
60	60	1	1	Cassebeer, Mrs. Henry Arthur Jr (Eleanor Genevieve Fosdick)	female	?	0	0	17770	27.7208	?	C	New
689	689	3	0	Braund, Mr. Lewis Richard	male	29	1	0	3460	7.0458	?	S	Bridg
1162	1162	3	0	Rush, Mr. Alfred George John	male	16	0	0	A/4. 20589	8.05	?	S	?
718	718	3	0	Chronopoulos, Mr. Demetrios	male	18	1	0	2680	14.4542	?	C	Gree
594	594	2	1	West, Mrs. Edwy Arthur (Ada Mary Worth)	female	33	1	2	C.A. 34651	27.75	?	S	Bour
1243	1243	3	0	Thomas, Mr. John	male	?	0	0	2681	6.4375	?	C	?
589	589	2	1	Wells, Miss. Joan	female	4	1	1	29103	23	?	S	Corn
879	879	3	0	Ilmakangas, Miss. Pieta Sofia	female	25	1	0	STON/O2. 3101271	7.925	?	S	?
743	743	3	1	Dahl, Mr. Karl Edwart	male	45	0	0	7598	8.05	?	S	Austi

```
#3. Cleansing the dataset
library(dplyr)
clean_titan = titan %>% select(c(-1,-4,-9,-11,-13))
```

Roll No: 17E111

glimpse(clean_titan) #To check the datatype/class of fields

```
Observations: 1,309
Variables: 8
$ pclass   <int> 2, 3, 2, 3, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3, 1, 3, 1, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
$ survived <int> 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
$ sex      <fct> male, male, female, female, female, female, male, male, male, female, male, female,
$ age      <fct> 24, 34, 19, 47, 22, ?, 29, 16, 18, 33, ?, 4, 25, 45, 49, ?, ?, 11, 34, 29, 39, 31,
$ sibsp    <int> 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
$ parch    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 2, 0, 1, 5, 0, 0, 0, 0, 0, 0, 0,
$ fare     <fct> 10.5, 8.05, 26, 14.5, 9.8375, 27.7208, 7.0458, 8.05, 14.4542, 27.75, 6.4375, 23, 7.
$ embarked <fct> S, S, S, S, S, C, S, S, C, S, C, S, S, S, C, C, S, S, S, S, Q, S, S, S, S, S, S, S,
```

```
clean_titan = titan %>% select(c(-1,-4,-9,-11,-13)) %>% mutate(pclass=factor(pclass,
                                        levels=c(1,2,3),
                                        labels=c("Upper","Middle","Lower")),
                                 survived=factor(survived,
                                        levels=c(0,1),
                                        labels=c("No","Yes")))
```

View(clean_titan)

24092019.R × | 19092019.R* × | clean_titan ×

Filter

	pclass	survived	sex	age	sibsp	parch	fare	embarked
1	Middle	Yes	male	24	0	0	10.5	S
2	Lower	No	male	34	0	0	8.05	S
3	Middle	Yes	female	19	0	0	26	S
4	Lower	No	female	47	1	0	14.5	S
5	Lower	No	female	22	0	0	9.8375	S
6	Upper	Yes	female	?	0	0	27.7208	C
7	Lower	No	male	29	1	0	7.0458	S
8	Lower	No	male	16	0	0	8.05	S
9	Lower	No	male	18	1	0	14.4542	C
10	Middle	Yes	female	33	1	2	27.75	S
11	Lower	No	male	?	0	0	6.4375	C
12	Middle	Yes	female	4	1	1	23	S
13	Lower	No	female	25	1	0	7.925	S

#4. Splitting the dataset into training and testing

```
library(caTools)
split=sample.split(clean_titan$survived,SplitRatio=0.8)
training_set = subset(clean_titan,split==TRUE)
testing_set = subset(clean_titan,split==FALSE)
```

training_set

```
> training_set
   pclass survived    sex  age sibsp parch     fare embarked
1  Middle      Yes   male   24     0     0     10.5        S
2   Lower       No   male   34     0     0     8.05        S
3  Middle      Yes female   19     0     0       26        S
4   Lower       No female   47     1     0     14.5        S
5   Lower       No female   22     0     0   9.8375        S
6   Upper      Yes female    ?     0     0  27.7208        C
8   Lower       No   male   16     0     0     8.05        S
9   Lower       No   male   18     1     0  14.4542        C
11  Lower       No   male    ?     0     0   6.4375        C
12 Middle      Yes female    4     1     1       23        S
```

Roll No: 17E111

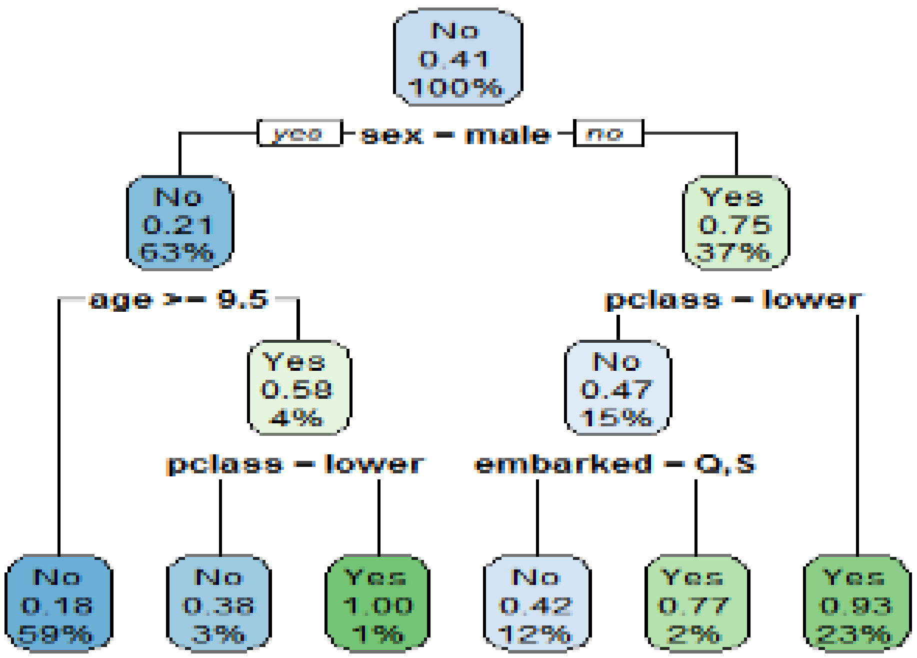
\
testing_set

```
> testing_set
   pclass survived    sex  age sibsp parch     fare embarked
7   Lower       No   male   29     1     0   7.0458        S
10 Middle      Yes female   33     1     2    27.75        S
20  Lower       No female   29     1     1  10.4625        S
23  Lower       No   male   38     0     0     7.05        S
25  Lower       No   male    ?     1     0  19.9667        S
28  Lower       No   male   16     0     0      9.5        S
33  Lower       No   male    ?     0     0     7.05        S
36  Lower       No   male   30     0     0     7.25        S
41  Upper      Yes female   49     0     0  25.9292        S
45  Upper      Yes female   58     0     0 146.5208        C
46  Upper       No   male   41     1     0  51.8625        S
68  Lower       No female   35     0     0     7.75        Q
72  Lower      Yes female   30     0     0   12.475        S
73  Lower       No   male   27     0     0   8.6625        S
```

#5. Buliding the Model
install.packages("rpart")
install.packages("rpart.plot")

library(rpart)
library(rpart.plot)

fit=rpart(formula = survived ~ .,data = training_set,method = "class")
rpart.plot(fit)



#6. Prediction making
predict_unseen=predict(object=fit,newdata=testing_set,type='class')
predict_unseen

Roll No: 17E111

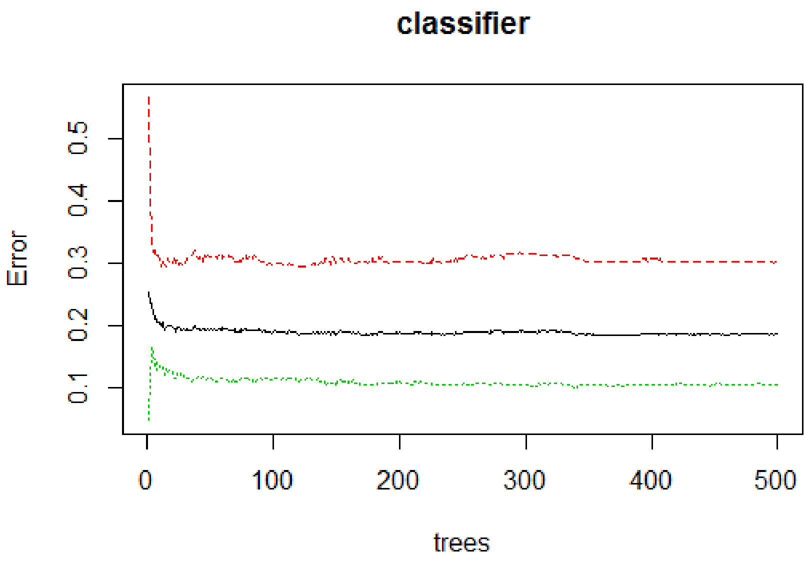
```
> predict_unseen
   7   10   20   23   25   28   33   36   41   45   46   68   72   73   83   86   90   92   99  107 
  No  Yes   No   No   No   No   No   No  Yes  Yes   No  Yes  Yes   No   No   No  Yes   No   No   No 
 133  142  149  152  160  163  164  180  184  188  189  194  198  204  217  231  236  243  246  251 
  No   No   No   No  Yes   No   No  Yes   No   No  Yes   No  Yes   No   No  Yes   No  Yes   No  Yes 
 283  284  288  297  299  321  328  347  352  354  360  364  365  373  375  393  395  402  404  406 
 Yes   No  Yes   No   No   No   No  Yes   No   No   No  Yes   No   No   No   No   No  Yes   No   No 
 420  422  426  433  434  436  441  445  448  449  452  456  459  463  474  487  495  502  503  505 
  No   No  Yes  Yes   No   No  Yes  Yes   No  Yes   No   No  Yes   No   No  Yes   No  Yes  Yes   No 
 537  545  546  562  569  571  573  580  582  588  590  606  607  608  614  617  618  628  629  638 
  No   No   No  Yes   No   No   No   No  Yes   No   No  Yes   No   No   No   No   No   No   No   No 
 655  686  688  696  698  699  706  713  714  721  722  723  731  732  744  747  751  755  758  767 
  No   No   No   No   No   No  Yes   No  Yes   No   No  Yes   No   No   No   No   No  Yes  Yes  Yes 
 798  802  803  807  809  812  816  823  832  834  839  858  861  863  865  874  875  876  881  882 
  No   No  Yes   No   No   No   No   No   No  Yes   No  Yes  Yes  Yes  Yes   No   No   No  Yes   No 
 924  933  936  943  949  950  954  964  965  971  972  979  980  981  982  983  984  993  997 1004 
```

#7. Confucion Matrix

```
cm=table(testing_set$survived,predict_unseen)
cm
sum(diag(prop.table(cm))) #Accuracy
```

```
> cm
     predict_unseen
       No Yes
  No  144  18
  Yes  37  63
> sum(diag(prop.table(cm))) #Accuracy
[1] 0.7900763
```

```
install.packages("caret",dependencies = T)
install.packages("randomForest",dependencies = T)
library(ggplot2)
library(lattice)
library(caret)
library(randomForest)
classifier = randomForest( x=training_set[-2],y=training_set$survived,ntree = 500)
y_pred=predict(classifier,newdata=test_set)
y_pred
plot(classifier)
```



Result:

Thus, the Decision Tree of Titanic Dataset has been constructed successfully.

Roll No: 17E111

Ex. No: 10
Date: 23-09-2019

Association Mining in R

Aim:

To perform the Association Mining for Market Basket Analysis dataset in R.

Procedure:

1. Open R studio.
2. In the script window, load the dataset required and perform Aprori algorithm.
3. After performing all the operations, Click on save button to save the document in local disk.
4. Exit from R studio.

Program:

```
#Aprori algorithm
install.packages("arules")
library(arules)
dataset = read.csv("Market_Basket_Optimisation.csv", header = F)
View(dataset)
```

	V1	V2	V3	V4	V5	V6	V7
1	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams
2	burgers	meatballs	eggs				
3	chutney						
4	turkey	avocado					
5	mineral water	milk	energy bar	whole wheat rice	green tea		
6	low fat yogurt						
7	whole wheat pasta	french fries					
8	soup	light cream	shallot				
9	frozen vegetables	spaghetti	green tea				

```
#Sparse Matrix
dataset = read.transactions("Market_Basket_Optimisation.csv", sep=",", rm.duplicates = TRUE)
```

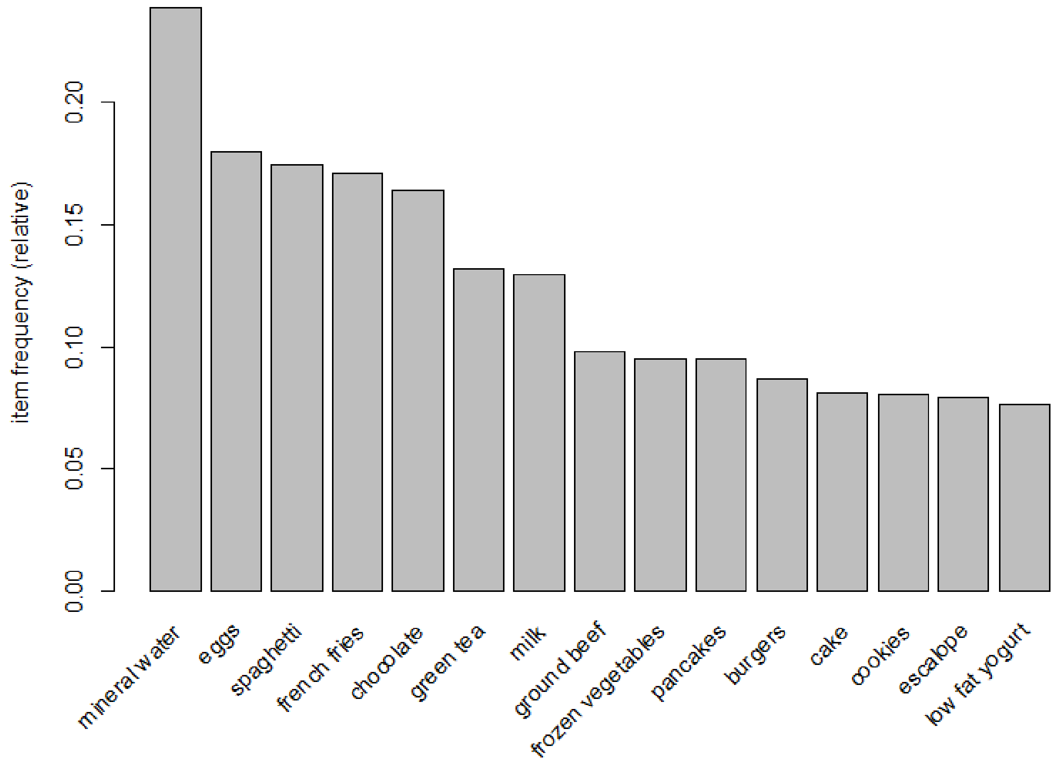
```
> dataset = read.transactions("Market_Basket_Optimisation.csv",
+                              sep=",",
+                              rm.duplicates = TRUE)
distribution of transactions with duplicates:
1
5
```

```
summary(dataset)
```

```
> summary(dataset)
transactions as itemMatrix in sparse format with
 7501 rows (elements/itemsets/transactions) and
 119 columns (items) and a density of 0.03288973

most frequent items:
mineral water          eggs     spaghetti  french fries     chocolate       (Other)
         1788          1348          1306          1282          1229         22405

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   18   19   20
1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17    4    1    2    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.914   5.000  20.000

includes extended item information - examples:
            labels
1          almonds
2 antioxydant juice
3        asparagus
```

Roll No: 17E111

itemFrequencyPlot(dataset, topN=15)



#Training Apriori on the dataset

rules = apriori(data=dataset, parameter=list(support= 0.004, confidence= 0.2))

```
> rules = apriori(data=dataset, parameter=list(support= 0.004, confidence= 0.2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.2    0.1    1 none FALSE            TRUE       5   0.004      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 30

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
sorting and recoding items ... [114 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [811 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
>
```

#Visualising the result

inspect(sort(rules, by='lift')[1:10])

```
> inspect(sort(rules, by='lift')[1:10])
     lhs                                     rhs             support     confidence lift     count
[1]  {light cream}                        => {chicken}       0.004532729 0.2905983  4.843951 34
[2]  {pasta}                              => {escalope}      0.005865885 0.3728814  4.700812 44
[3]  {pasta}                              => {shrimp}        0.005065991 0.3220339  4.506672 38
[4]  {eggs,ground beef}                   => {herb & pepper} 0.004132782 0.2066667  4.178455 31
[5]  {whole wheat pasta}                  => {olive oil}     0.007998933 0.2714932  4.122410 60
[6]  {herb & pepper,spaghetti}            => {ground beef}   0.006399147 0.3934426  4.004360 48
[7]  {herb & pepper,mineral water}        => {ground beef}   0.006665778 0.3906250  3.975683 50
[8]  {tomato sauce}                       => {ground beef}   0.005332622 0.3773585  3.840659 40
[9]  {mushroom cream sauce}               => {escalope}      0.005732569 0.3006993  3.790833 43
[10] {frozen vegetables,mineral water,spaghetti} => {ground beef}   0.004399413 0.3666667  3.731841 33
```

Result:

Thus, the Association Mining for Market Basket Analysis dataset has been analysed successfully.

Roll No: 17E111

Ex. No: 11
Date: 25-09-2019

K-Means Clustering in R

Aim:

To perform the K-Means clustering for the given dataset.

Procedure:

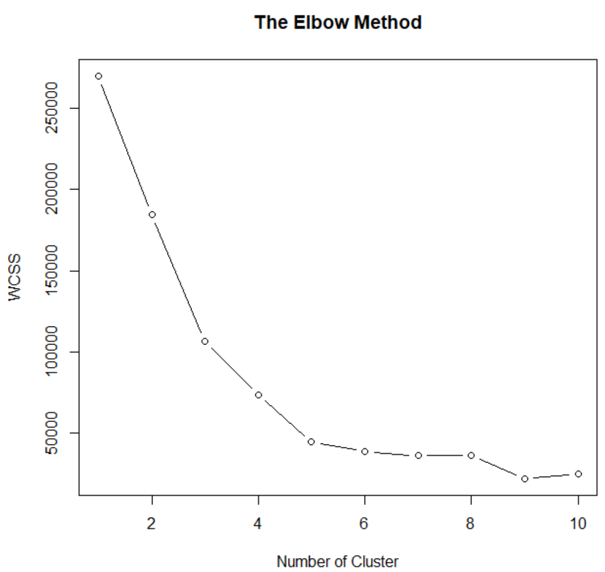
1. Open R studio.
2. In the script window, load the dataset required and perform Aprori algorithm.
3. After performing all the operations, Click on save button to save the document in local disk.
4. Exit from R studio.

Program:

```
#K-Means Clustering
#Import the dateset
dataset = read.csv("Mall_Customers.csv")
View(dataset)
```

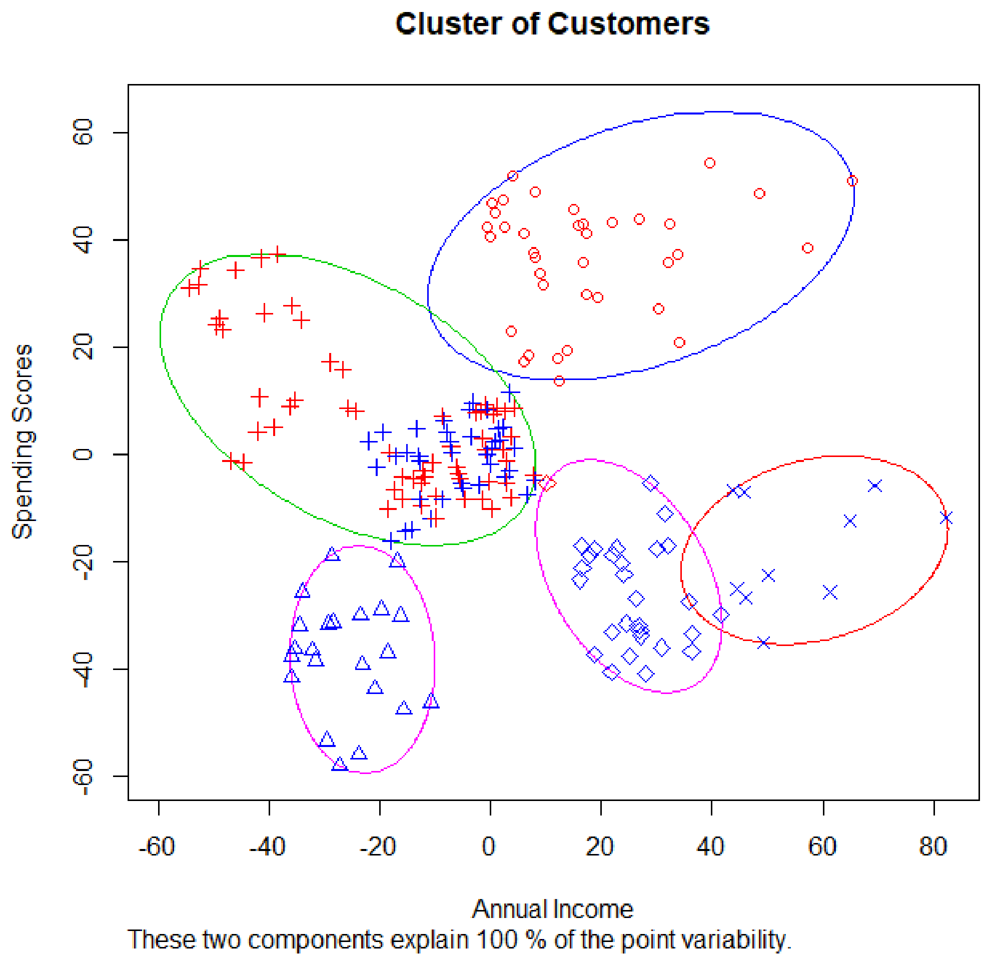
	CustomerID	Genre	Age	Annual.Income..k..	Spending.Score..1.100.
1	1	Male	19	15	39
2	2	Male	21	15	81
3	3	Female	20	16	6
4	4	Female	23	16	77
5	5	Female	31	17	40
6	6	Female	22	17	76
7	7	Female	35	18	6
8	8	Female	23	18	94
9	9	Male	64	19	3
10	10	Female	30	19	72

```
dataset = dataset[4:5]

#Elbow plot
library(cluster)
set.seed(500)
wcss = vector()
for(i in 1:10)
  wcss[i]=sum(kmeans(dataset,i)$withinss)
plot(1:10, wcss, type='b', main="The Elbow Method", xlab="Number of Cluster", ylab='WCSS')
```



Roll No: 17E111

```
#Fitting K-mean to the dataset
kmeans=kmeans(x=dataset,centers = 5 )
y_kmeans = kmeans$cluster

#Visualising the cluster
clusplot(dataset, y_kmeans,col.p = c("red","blue"), lines = 0, color = T, main="Cluster of Customers",
                    xlab = "Annual Income", ylab = "Spending Scores")
```



Result:

Thus, the K-Means clustering has been implemented successfully.

Roll No: 17E111

Ex. No: 12
Date: 30-09-2019

Analysis of Variance in R

Aim:

To analyse the poisons dataset by the concept of ANOVA in R.

Procedure:

1. Open R studio.
2. In the script window, load the dataset required and perform ANOVA operations.
3. After performing all the operations, Click on save button to save the document in local disk.
4. Exit from R studio.

Program:

```
library(dplyr)
PATH = "https://raw.githubusercontent.com/guru99-edu/R-Programming/master/poisons.csv"
ds = read.csv(PATH) %>% select(-X) %>% mutate(poison=factor(poison, ordered = T))
View(ds)
```

	time	poison	treat
1	0.31	1	A
2	0.45	1	A
3	0.46	1	A
4	0.43	1	A
5	0.36	2	A
6	0.29	2	A
7	0.40	2	A
8	0.23	2	A
9	0.22	3	A
10	0.21	3	A

```
str(ds)
```

```
> str(ds)
'data.frame':	48 obs. of  3 variables:
 $ time  : num  0.31 0.45 0.46 0.43 0.36 0.29 0.4 0.23 0.22 0.21 ...
 $ poison: Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 1 1 2 2 2 2 3 3 ...
 $ treat : Factor w/ 4 levels "A","B","C","D": 1 1 1 1 1 1 1 1 1 1 ...
```

```
#ANOVA Step1
levels(ds$poison)
```

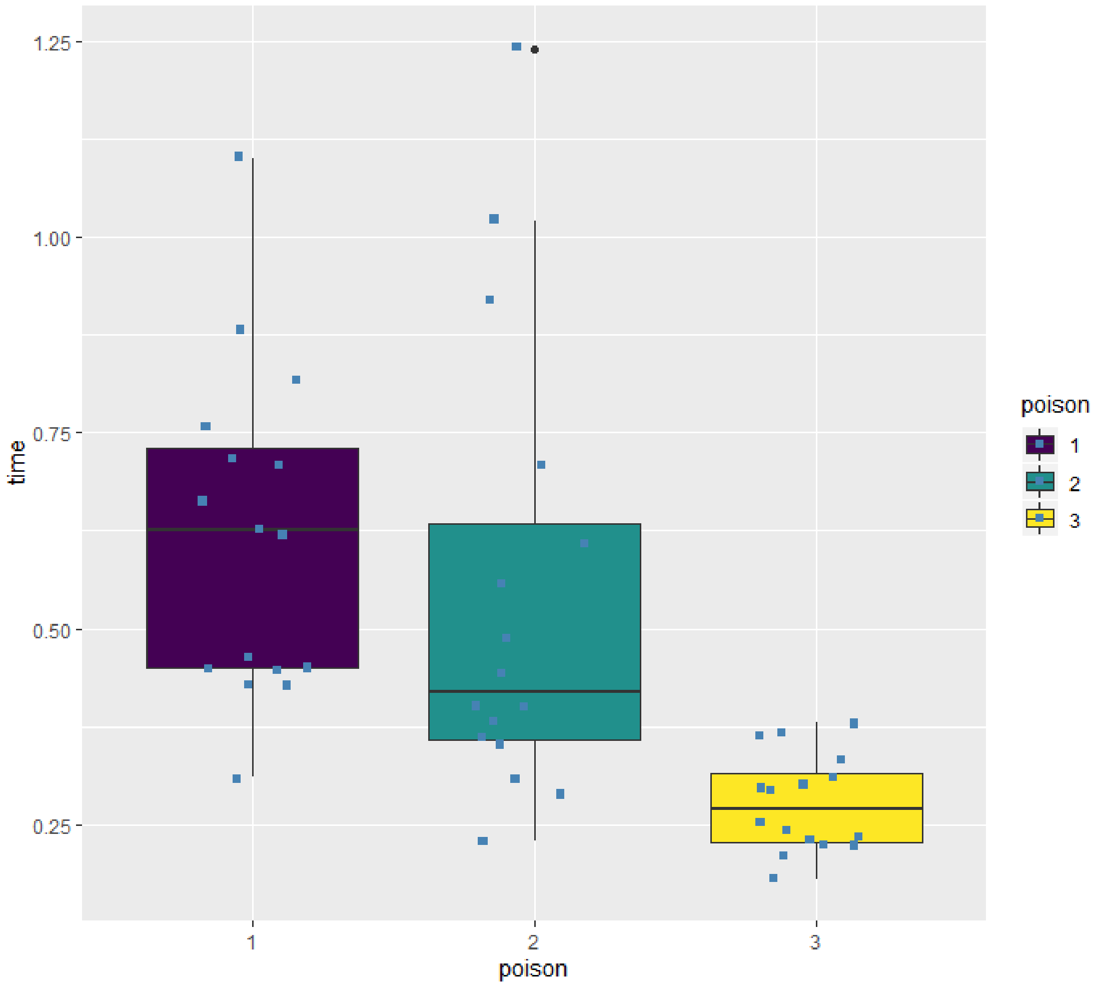
```
> #ANOVA Step1
> levels(ds$poison)
[1] "1" "2" "3"
```

```
#Step 2: Group by poison
ds %>% group_by(poison) %>% summarise(count_poison = n(), mean_time = mean(time, na.rm=T),
                                      sd_time= sd(time, na.rm=T))
```

Roll No: 17E111

```
> #Step 2: Group by poison
> ds %>% group_by(poison) %>% summarise(count_poiso
+                                        sd_time= sd
# A tibble: 3 x 4
  poison count_poison mean_time sd_time
  <ord>         <int>     <dbl>   <dbl>
1 1                16     0.618  0.209
2 2                16     0.544  0.289
3 3                16     0.276  0.0623
> 
```

#Step 3: Box plot
library(ggplot2)
ggplot(ds,aes(x=poison, y=time, fill=poison)) + geom_boxplot() + geom_jitter(shape=15, color='steelblue',
position = position_jitter(0.21))



#Step 4: One way ANOVA
anova_one_way = aov(formula = time ~ poison, data = ds)
summary(anova_one_way)

```
> anova_one_way = aov(formula = time ~ poison, data = ds)
> summary(anova_one_way)
            Df Sum Sq Mean Sq F value   Pr(>F)    
poison       2  1.033  0.5165   11.79 7.66e-05 ***
Residuals   45  1.972  0.0438                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Roll No: 17E111

#Pairwise Comparision
TukeyHSD(anova_one_way)

```
> #Pairwise Comparision
> TukeyHSD(anova_one_way)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = time ~ poison, data = ds)

$poison
         diff        lwr         upr     p adj
2-1 -0.073125 -0.2525046  0.10625464 0.5881654
3-1 -0.341250 -0.5206296 -0.16187036 0.0000971
3-2 -0.268125 -0.4475046 -0.08874536 0.0020924
```

#Step 5 : Two-way ANOVA
anova_two_way <- aov(formula =time ~ poison + treat, data =ds)
summary(anova_two_way)

```
> #Step 5 : Two-way ANOVA
> anova_two_way <- aov(formula =time ~ poison + treat, data =ds)
> summary(anova_two_way)
            Df Sum Sq Mean Sq F value  Pr(>F)    
poison       2 1.0330  0.5165   20.64 5.7e-07 ***
treat        3 0.9212  0.3071   12.27 6.7e-06 ***
Residuals   42 1.0509  0.0250                    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Result:

Thus, the Analysis of Variance (ANOVA) has been done successfully for the given dataset