Title: - Data Analysis in R

Aim: - To perform data analysis using R programming.

Lab Objectives: -

Students will understand following R programming concepts:

I. Regression Technique
II. Market basket analysis using Apriori algorithm
III. Naïve Bayes Classification
IV. K means Clustering

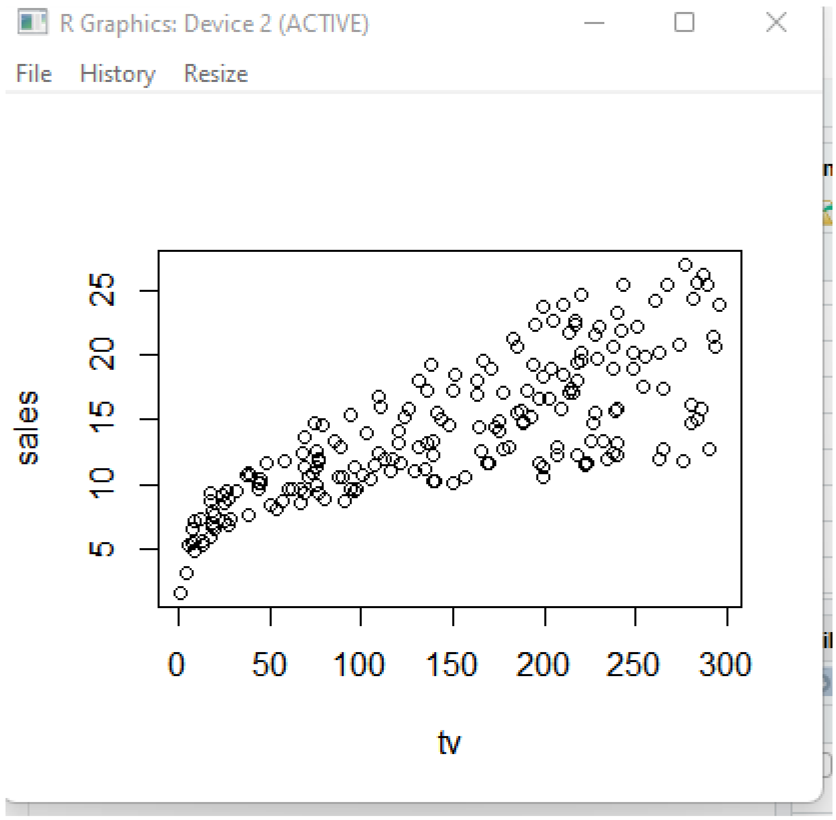
I. Linear Regression in R

- Regression analysis is a very widely used statistical tool to establish a relationship model between two variables.
- One of these variable is called predictor variable whose value is gathered through experiments.
- The other variable is called response variable whose value is derived from the predictor variable.
- Linear regression is used to predict the value of an outcome variable *Y* based on one or more input predictor variables *X*.
- Mathematically a linear relationship represents a straight line when plotted as a graph.
- The general mathematical equation for a linear regression is −
- $\boldsymbol{y} = \mathbf{b0} + \mathbf{b1} * \boldsymbol{x}$
- Following is the description of the parameters used −
  - y is the response variable.
  - x is the predictor variable.
  - b1 – slope
  - b0 - intercept
  - Collectively, they are called *regression coefficients*.
- For example, we want to predict weight (y) from height (x), the linear regression model can be represented by the following equation
- Weight= b0 + b1 * height
  - b1 is called slope because it defines the slope of the line or how x translates into a y i.e by how much y is affected by change in x
- The goal is to find best estimates for the coefficients to minimize the error in predicting y from x

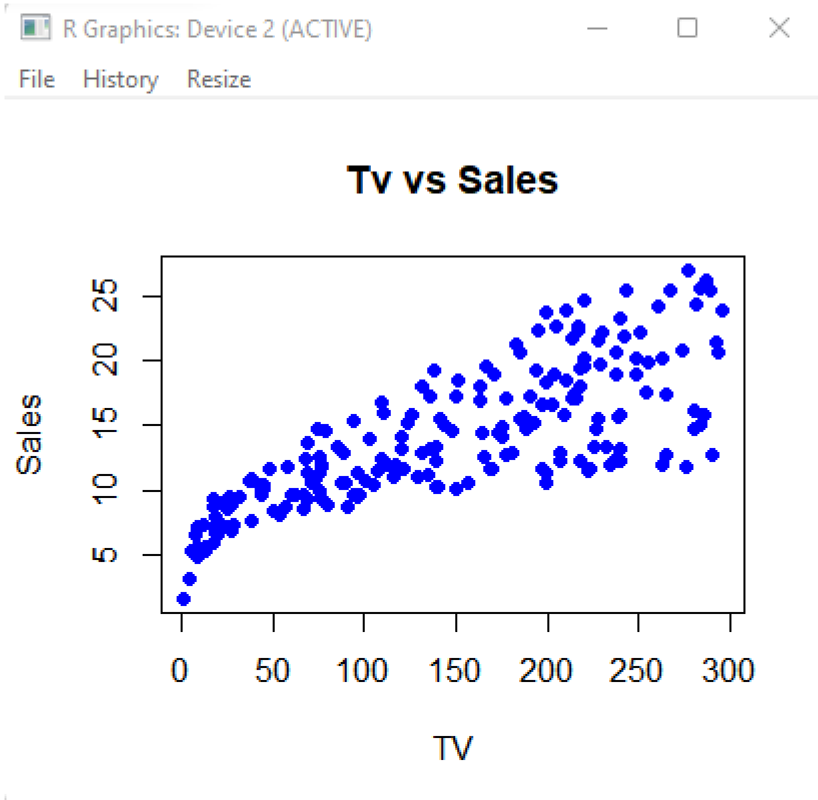
Download marshall adv. csv Dataset to perform operations

```
200 NA 232.1    8.6        8.7  13.4
> data<-read.csv("advr.csv")
> data
       TV Radio Newspaper Sales
1   230.1  37.8      69.2  22.1
2    44.5  39.3      45.1  10.4
3    17.2  45.9      69.3   9.3
4   151.5  41.3      58.5  18.5
5   180.8  10.8      58.4  12.9
6     8.7  48.9      75.0   7.2
7    57.5  32.8      23.5  11.8
8   120.2  19.6      11.6  13.2
9     8.6   2.1       1.0   4.8
10  199.8   2.6      21.2  10.6
```

```
tv<-data$TV
tv
sales<data$Sales
sales
```



plot(tv,sales,pch=16,cex=1,col='blue',main='Tv vs Sales',xlab = 'TV',ylab = 'Sales')



model<-lm

summary(model)

```
> model<-lm(sales~tv)
> summary(model)

Call:
lm(formula = sales ~ tv)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.3860 -1.9545 -0.1913  2.0671  7.2124 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 7.032594   0.457843   15.36   <2e-16 ***
tv          0.047537   0.002691   17.67   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.259 on 198 degrees of freedom
Multiple R-squared:  0.6119,    Adjusted R-squared:  0.6099 
F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16

> |
```

attributes(model)

```
> attributes(model)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values" "assign"        "qr"
 [8] "df.residual"   "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"
```

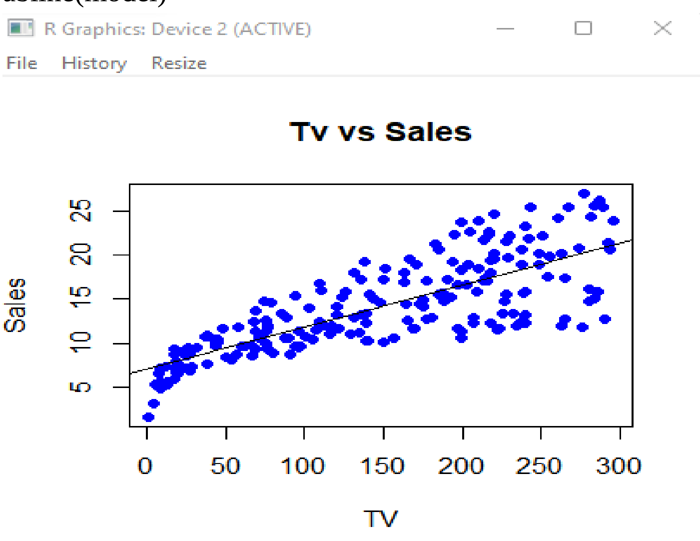
coefficients(model)

```
> coefficients(model)
(Intercept)          tv
 7.03259355  0.04753664
>
```


```
> coefficients(model)
(Intercept)          tv
 7.03259355  0.04753664
>
```


```
> coef(model)
(Intercept)          tv
 7.03259355  0.04753664
>
```

abline(model)



II. Market Basket Analysis in R

- ▷ The increasing volume of data and the growing importance of retail analytics made it easy for retailers to know their customers better.
- ▷ Data can help retailers to understand customer behavior, plan and promote products, increase sales, improve customer experience, and optimize supply chain performance.
- ▷ There are many algorithms and techniques used in retail that help uncover better insights and predict future events.
- ▷ One of the key and widely used techniques in retail is Market Basket Analysis.
- ▷ It works by searching for combinations of items that often happen in transactions together.
- ▷ Market Basket Analysis is a technique that is used to discover the association between items.
- ▷ In simplest terms, it allows retailers to identify a relationship between items that generally people buy together.
- ▷ For instance, if one person buys 'bread', he/she more likely to buy 'butter' or 'jam' which is predicted as a 'go-along' item with the purchase
- ▷ To implement this, associate rule mining is used.
- ▷ Association Rule Mining is a rule-based machine learning method to find associations and relationships between large sets of items.

- ▷ **Support:**
  - ○ This is one of the important measures to determine how frequently an itemset occurs in the transaction as a percentage of all transactions.
  - ○ Support is the number of transactions that include both {A} and {B} parts as a percentage of the total number of transactions.

$$\mathbf{Support} = \frac{\mathbf{(A + B)}}{\mathbf{Total}}$$

- ▷ **Confidence:**
  - ○ This rule is the ratio of the number of transactions that include items in {A} and {B} to the number of transactions that include items in {A}.
  - ○ It can be understood as to how often items in B appear in transactions that contain A only. It is a conditional probability.

$$\mathbf{Confidence} = \frac{\mathbf{(A + B)}}{\mathbf{A}}$$

- ▷ **Lift:**
  - ○ This third measure, lift or lift ratio is the ratio of confidence to expected confidence.
  - ○ We can say that this rule shows us how much better a rule is at predicting the result than just assuming it.
  - ○ Greater lift value tells how strong the association is.
  - ○ It shows us the rate of confidence that B will be purchased given that A was purchased.

- In other way Lift = Confidence(A=>B) / Support(B)

$$\textbf{Lift} = \left(\frac{\left(\frac{(A+B)}{A}\right)}{\left(\frac{B}{Total}\right)} \right)$$

▷

install.packages("arules")
install.packages("arulesViz")
library(arules)
library(arulesViz)
w1=read.table("E:/Adbms/comm.csv")

> trans=read.transactions("E:/Adbms/comm.csv",format="basket",sep=",");

itemFrequencyPlot(trans,topN=20,type="absolute")

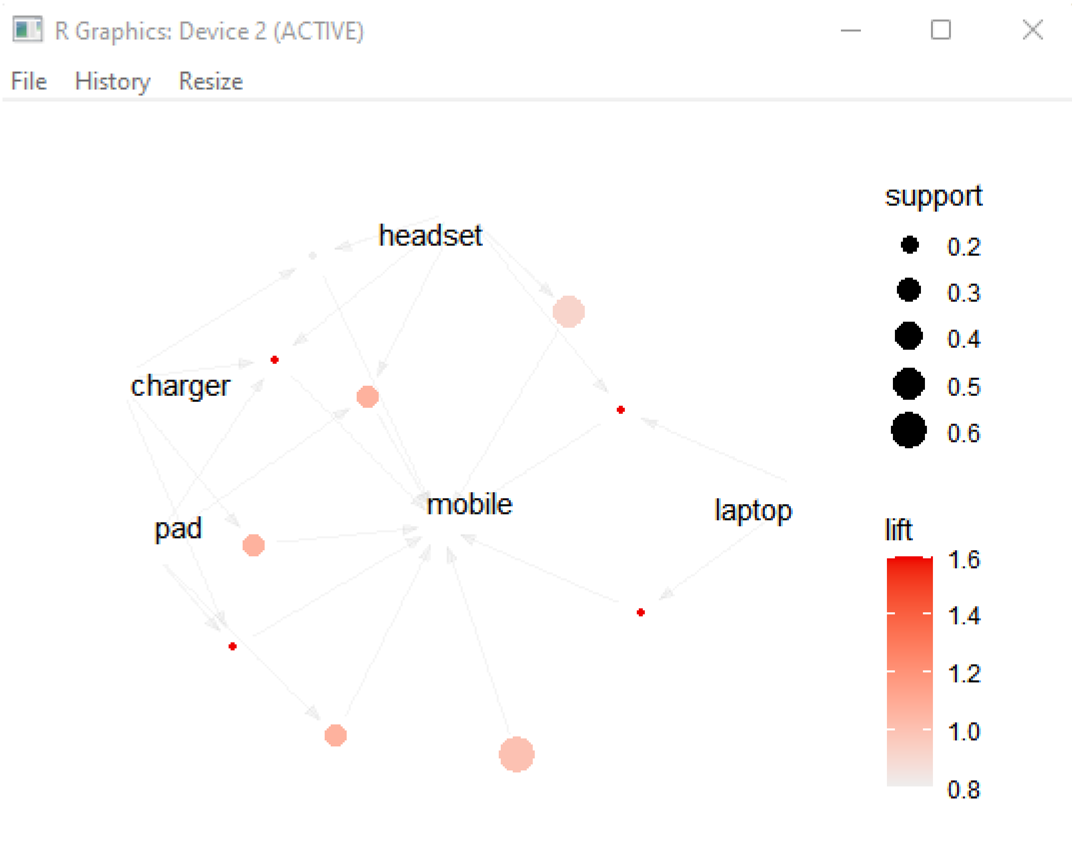
rules<-apriori(data=trans,parameter = list(supp=0.001,conf=0.08),appearance = list(default="lhs",rhs="mobile"),control=list(verbose=F))

rules<-sort(rules,decreasing = TRUE,by="confidence")

inspect(rules[1:10])

```
> rules<-sort(rules,decreasing = TRUE,by="confidence")
> inspect(rules[1:10])
     lhs                        rhs      support confidence coverage lift      count
[1]  {laptop}                => {mobile} 0.125   1.0000000  0.125    1.6000000 1
[2]  {headset, laptop}       => {mobile} 0.125   1.0000000  0.125    1.6000000 1
[3]  {charger, pad}          => {mobile} 0.125   1.0000000  0.125    1.6000000 1
[4]  {charger, headset, pad} => {mobile} 0.125   1.0000000  0.125    1.6000000 1
[5]  {charger}               => {mobile} 0.250   0.6666667  0.375    1.0666667 2
[6]  {pad}                   => {mobile} 0.250   0.6666667  0.375    1.0666667 2
[7]  {headset, pad}          => {mobile} 0.250   0.6666667  0.375    1.0666667 2
[8]  {}                      => {mobile} 0.625   0.6250000  1.000    1.0000000 5
[9]  {headset}               => {mobile} 0.500   0.5714286  0.875    0.9142857 4
[10] {charger, headset}      => {mobile} 0.125   0.5000000  0.250    0.8000000 1
> 
```

plot(rules,method = "graph")



IV. Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Likelihood

Class Prior Probability

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Posterior Probability

Predictor prior probability

Working of Naïve Bayes' Classifier:

- Working of Naïve Bayes' Classifier can be understood with the help of the below example:
- Suppose we have a dataset of weather conditions and corresponding target variable "Play".
- So, using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions.
- So, to solve this problem, we need to follow the below steps:
  - Construct a frequency table for each attribute against the target.
  - Transform the frequency tables to likelihood tables
  - Finally use the Naive Bayesian equation to calculate the posterior probability for each class.
  - The class with the highest posterior probability is the outcome of prediction.

Naive bayes

```
rm(list = ls())

NBdataset<-read.table("new_dataset.csv",header = TRUE,sep = ",")

install.packages("e1071")
library(e1071)

classifier<-naiveBayes(NBdataset[,1:4],NBdataset[,5])
table(predict(classifier,NBdataset[,5]),NBdataset[,5],dnn=list('predicted','actual'))

classifier$tables


NBdataset[15,-5]<-
as.factor(c(Outlook="Sunny",Temperature="Cool",Humidity="High",wind="Strong"))
print(NBdataset[15,-5])
result<-predict(classifier,NBdataset[15,-5])
print(result)
```


```
rm(list = ls())

NBdataset<-read.table("new_dataset.csv",header = TRUE,sep = ",")

install.packages("e1071")
library(e1071)

classifier<-naiveBayes(NBdataset[,1:4],NBdataset[,5])
table(predict(classifier,NBdataset[,5]),NBdataset[,5],dnn=list('predicted','actual'))
```

```
> table(predict(classifier,NBdataset[,5]),NBdataset[,5],dnn=list('predicted','actual'))
         actual
predicted no yes
      no   0   0
      yes  5   9
```

```
> classifier$tables
$Outlook
                Outlook
NBdataset[, 5]  Overcast     Rainy     Sunny
           no  0.0000000 0.6000000 0.4000000
           yes 0.4444444 0.2222222 0.3333333

$Temp
                Temp
NBdataset[, 5]      Cool       Hot      Mild
           no  0.2000000 0.4000000 0.4000000
           yes 0.3333333 0.2222222 0.4444444

$Humidity
                Humidity
NBdataset[, 5]      High    Normal
           no  0.8000000 0.2000000
           yes 0.3333333 0.6666667

$Windy
                Windy
NBdataset[, 5]         f         t
           no  0.4000000 0.6000000
           yes 0.6666667 0.3333333

>
```

NBdataset[15,-5]<-
as.factor(c(Outlook="Sunny",Temperature="Cool",Humidity="High",wind="Strong"))
print(NBdataset[15,-5])

```
> NBdataset[15,-5]<-as.factor(c(Outlook="Sunny",Temperature="Cool"
> print(NBdataset[15,-5])
   Outlook Temp Humidity  Windy
15   Sunny Cool     High Strong
>
```

result<-predict(classifier,NBdataset[15,-5])
print(result)

```
> print(result)
[1] yes
Levels: no yes
>
```

Exercises

1. Write a program to perform k means clustering on USArrest dataset. Perform data pre-processing if required.

```
View(USArrests)

mydata<-select(USArrests,c(1,2,3,4))

View(mydata)
```

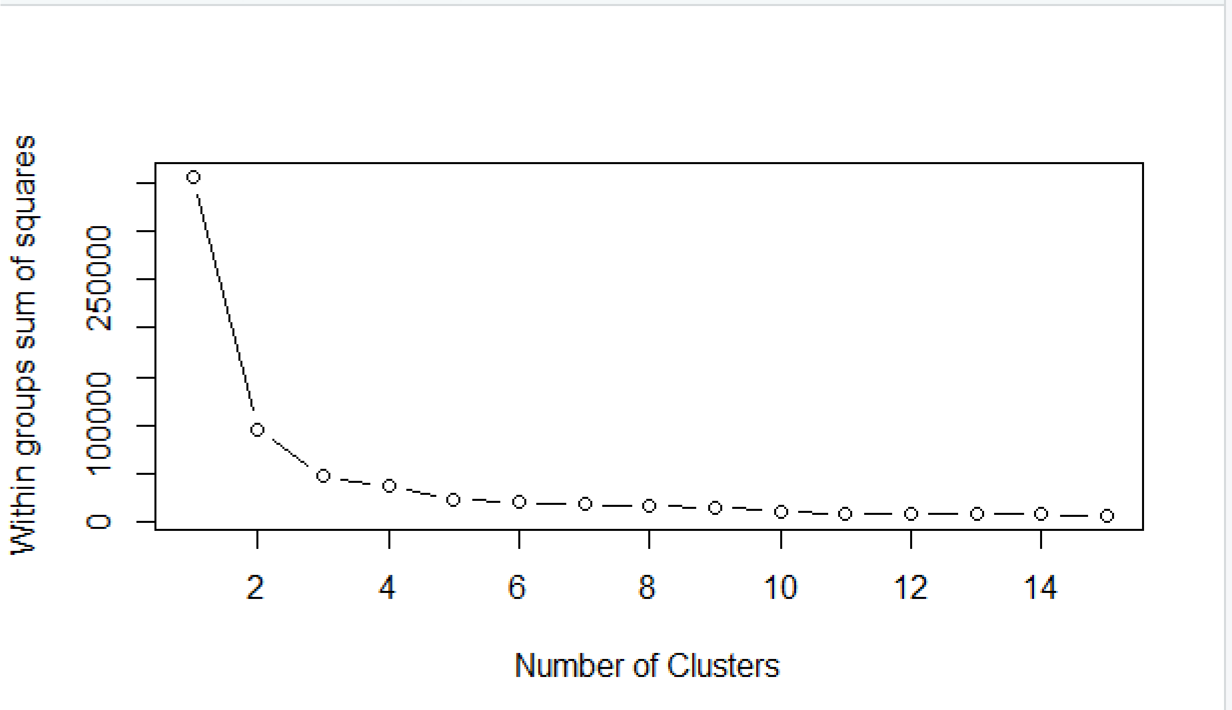
Untitled1 × K Means.R* × mydata × USArrests ×

Filter

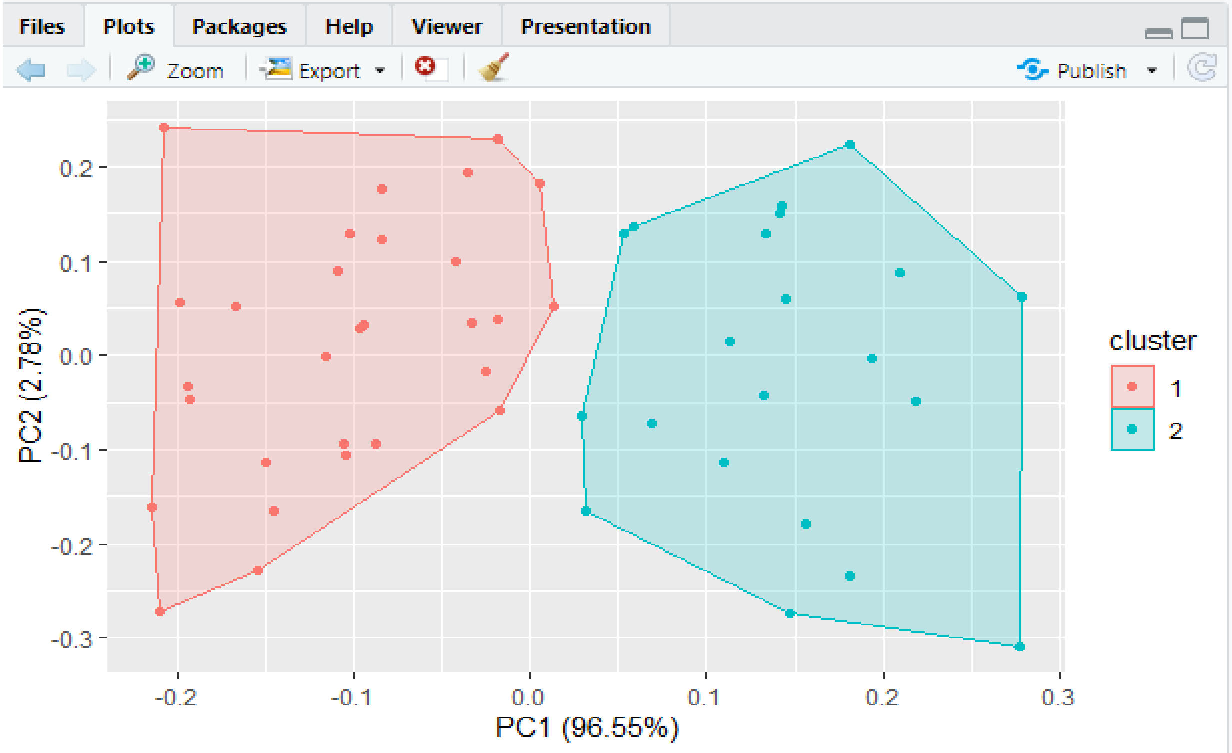
	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7
Connecticut	3.3	110	77	11.1
Delaware	5.9	238	72	15.8
Florida	15.4	335	80	31.9
Georgia	17.4	211	60	25.8
Hawaii	5.3	46	83	20.2

```
wssplot <- function(data, nc=15, seed=1234){

 wss <- (nrow(data)-1)*sum(apply(data,2,var))

 for (i in 2:nc){

  set.seed(seed)

  wss[i] <- sum(kmeans(data, centers=i)$withinss)}

 plot(1:nc, wss, type="b", xlab="Number of Clusters",

     ylab="Within groups sum of squares")
```

```
  wss
}
wssplot(mydata)
```




```
KM=kmeans(mydata,2)
autoplot(KM,mydata,frame=TRUE)
```



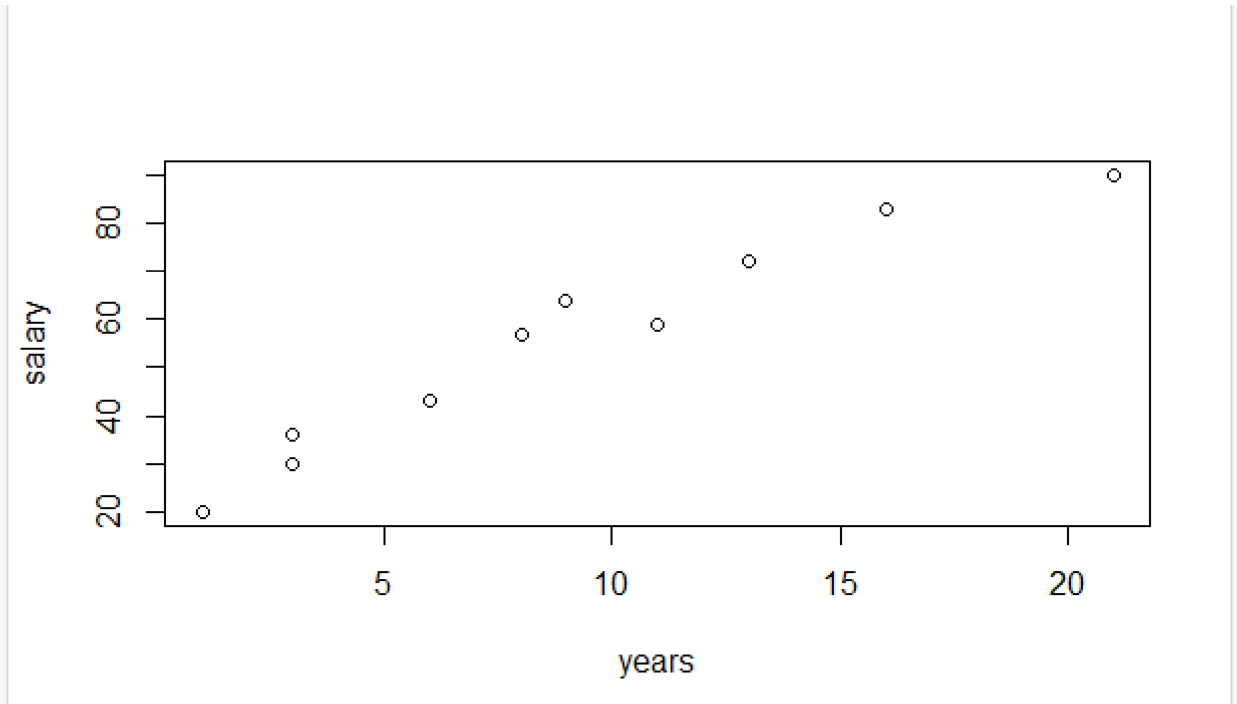

```
> KM$centers
     Murder  Assault UrbanPop     Rape
1  4.841379 109.7586 64.03448 16.24828
2 11.857143 255.0000 67.61905 28.11429
>
```

2. Implement Regression Classification for following example using R

years=(3,8,9,13,3,6,11,21,1,16)

salary=(30,57,64,72,36,43,59,90,20,83)

Predict salary of a person having 10 years of experience in a company.




```
> years <- c(3, 8, 9, 13, 3, 6, 11, 21, 1, 16)
> salary<-c(30,57,64,72,36,43,59,90,20,83)
> plot(years,salary)
> data <- data.frame(years = years, salary = salary)
> model <- lm(salary ~ years, data = data)
> new_data <- data.frame(years = 10)
> predicted_salary <- predict(model, newdata = new_data)
> print(predicted_salary)
       1
58.58373
> 
```

3. Write a program to perform market basket analysis on Groceries dataset and display the top 5 important rules after sorting by confidence.

install.packages("arules")

library(arules)

rules <- apriori(Groceries, parameter = list(support = 0.001, confidence = 0.5))

```
> rules <- apriori(Groceries, parameter = list(support = 0.001, confidence = 0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen
        0.5    0.1    1 none FALSE            TRUE       5   0.001      1     10
 target  ext
  rules TRUE

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

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.05s].
writing ... [5668 rule(s)] done [0.02s].
creating S4 object  ... done [0.03s].
```

rules <- sort(rules, by = "confidence", decreasing = TRUE)

top_rules <- head(rules, 5)

inspect(top_rules)

```
> rules <- sort(rules, by = "confidence", decreasing = TRUE)
> top_rules <- head(rules, 5)
> inspect(top_rules)
    lhs                        rhs             support confidence    coverage     lift count
[1] {rice,
     sugar}                 => {whole milk} 0.001220132          1 0.001220132 3.913649    12
[2] {canned fish,
     hygiene articles}      => {whole milk} 0.001118454          1 0.001118454 3.913649    11
[3] {root vegetables,
     butter,
     rice}                  => {whole milk} 0.001016777          1 0.001016777 3.913649    10
[4] {root vegetables,
     whipped/sour cream,
     flour}                 => {whole milk} 0.001728521          1 0.001728521 3.913649    17
[5] {butter,
     soft cheese,
     domestic eggs}         => {whole milk} 0.001016777          1 0.001016777 3.913649    10
>
```

Exersise:

rm(list = ls())

getwd()

NBdataset<-read.table("input.csv",header = TRUE,sep = ",")

install.packages("e1071")

library(e1071)

classifier<-naiveBayes(NBdataset[,1:4],NBdataset[,5])

table(predict(classifier,NBdataset[,5]),NBdataset[,5],dnn=list('predicted','actual'))

```
> table(predict(classifier,NBdataset[,5]),NBdataset[,5],dnn=list('predicted','actual'))
         actual
predicted No Yes
      No   5   5
      Yes  0   0
```

classifier$tables

```
> classifier$tables
$Chills
              Chills
NBdataset[, 5]  No Yes
           No  0.4 0.6
           Yes 0.6 0.4

$Runny_nose
              Runny_nose
NBdataset[, 5]  No Yes
           No  0.8 0.2
           Yes 0.4 0.6

$Headache
              Headache
NBdataset[, 5] mild  No stong strong
           No   0.2 0.4   0.0    0.4
           Yes  0.4 0.2   0.2    0.2

$Fever
              Fever
NBdataset[, 5]  No Yes
           No  0.4 0.6
           Yes 0.4 0.6

>
```

NBdataset[15,-5]<-as.factor(c(Chills="Yes",Runny_nose="No",Headache="mild",fever="Yes"))

print(NBdataset[15,-5])

result<-predict(classifier,NBdataset[15,-5])

print(result)

```
> NBdataset[15,-5]<-as.factor(c(Chills="Yes",Runny_nose="No",Headache="mild",fever="Yes"))
> print(NBdataset[15,-5])
   Chills Runny_nose Headache Fever
15    Yes         No     mild   Yes
> result<-predict(classifier,NBdataset[15,-5])
> print(result)
[1] No
Levels: No Yes
>
```