S.I.W.S

N.R SWAMY COLLEGE OF COMMERCE & ECONOMICS AND SMT. THIRUMALAI COLLEGE OF SCIENCE.

Data Science

JAISWAR ROHIT MANOJ
T.Y.BSC C.S.
Roll No.: 39041

Year 2022-2023

S.I.W.S

N.R SWAMY COLLEGE OF COMMERCE AND ECONOMICS

AND

SMT. THIRUMALAI COLLEGE OF SCIENCE

Plot No.337, Major R. Parmeshwaran Marg, Sewree Wadala Estate,
Wadala, Mumbai-400 031.

T.Y.B.Sc.(Computer Science)

Semester VI

CERTIFICATE

Class:_________ University Seat No.:_________

Roll No.:___________

This is to certify that the experiments entered in this journal is the work of Mr./Ms.______________ in the Computer Science Department of S.I.W.S Degree College during the year 2022 – 2023.

_________________ Teacher-In-Charge

_________________ Co-Ordinator

_________________ Internal Examiner

_________________ External Examiner

Date: _________________

College Stamp

INDEX

PRACTICAL NO:- 1

Data curation and management for Unstructured data.

AIM:- Practical of Data collection, Data curation and management for Unstructured data.

DESCRIPTION:- Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes.

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

When people use the term “NoSQL database”, they typically use it to refer to any non-relational database. Some say the term “NoSQL” stands for “non SQL” while others say it stands for “not only SQL.” Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

A common misconception is that NoSQL databases or non-relational databases don’t store relationship data well. NoSQL databases can store relationship data—they just store it differently than relational databases do. In fact, when compared with SQL databases, many find modeling relationship data in NoSQL databases to be easier than in SQL databases, because related data doesn’t have to be split between tables.

NoSQL data models allow related data to be nested within a single data structure.

How NoSQL Databases Work:

One way of understanding the appeal of NoSQL databases from a design perspective is to look at how the data models of a SQL and a NoSQL database might look in an oversimplified example using address data.

The SQL Case. For an SQL database, setting up a database for addresses begins with the logical construction of the format and the expectation that the records to be stored are going to remain relatively unchanged. After analyzing the expected query patterns, an SQL database might optimize storage in two tables, one for basic information and one pertaining to being a customer, with last name being the key to both tables. Each row in each table is a single customer, and each column has the following fixed attributes:

OUTPUT:

```
test> show dbs
admin    40.00 KiB
config  108.00 KiB
local    40.00 KiB
taslim    8.00 KiB
test> use SIWS CLG
switched to db SIWS
SIWS> db.createCollection("DS")
{ ok: 1 }
SIWS> db.DS.insert({Dept:"cs",Name:"Taslim",Roll_no:"18",class:"TYCS"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63f1b2e60e10cf2978cab3c1") }
}
SIWS> db.DS.insert({Dept:"cs",Name:"Mohsin",Roll_no:"33",class:"TYCS"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63f1b3670e10cf2978cab3c2") }
}
SIWS> db.DS.find()
[
  {
    _id: ObjectId("63f1b2e60e10cf2978cab3c1"),
    Dept: 'cs',
    Name: 'Taslim',
    Roll_no: '18',
    class: 'TYCS'
  },
  {
    _id: ObjectId("63f1b3670e10cf2978cab3c2"),
    Dept: 'cs',
    Name: 'Mohsin',
    Roll_no: '33',
    class: 'TYCS'
  }
]
SIWS>
```

1. Last name :: first name :: middle initial :: address fields :: email address :: phone number
2. Last name :: date of birth :: account number :: customer years :: communication preferences
3. Each type of NoSQL database would be designed with a specific customer situation in mind, and there would be technical reasons for how each kind of database would be organized. The simplest type to describe is the document database, in which it would be natural to combine both the basic information and the customer information in one JSON document. In this case, each of the SQL column attributes would be fields and the details of a customer's record would be the data values associated with each field.

For example: Last_name: "Jones", First_name: "Mary", Middle_initial: "S"

CONCLUSION:- The above program has been executed successfully.

OUTPUT:

```
SIWS> db.DS.update({"Roll_no":"33"},{$set:{"Roll_no":"60"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
SIWS> db.DS.find().pretty()
[
  {
    _id: ObjectId("63f1b2e60e10cf2978cab3c1"),
    Dept: 'cs',
    Name: 'Taslim',
    Roll_no: '69',
    class: 'TYCS'
  },
  {
    _id: ObjectId("63f1b3670e10cf2978cab3c2"),
    Dept: 'cs',
    Name: 'Mohsin',
    Roll_no: '69',
    class: 'TYCS'
  },
  {
    _id: ObjectId("63f1b59cc286711c3279d274"),
    Dept: 'cs',
    Name: 'Taslim',
    Roll_no: '18',
    class: 'TYCS'
  },
  {
    _id: ObjectId("63f1b5afc286711c3279d275"),
    Dept: 'cs',
    Name: 'Mohsin',
    Roll_no: '60',
    class: 'TYCS'
  }
]
```

```
SIWS> db.DS.remove({"Roll_no":"60"})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
SIWS> db.DS.find().pretty()
[
  {
    _id: ObjectId("63f1b2e60e10cf2978cab3c1"),
    Dept: 'cs',
    Name: 'Taslim',
    Roll_no: '69',
    class: 'TYCS'
  },
  {
    _id: ObjectId("63f1b3670e10cf2978cab3c2"),
    Dept: 'cs',
    Name: 'Mohsin',
    Roll_no: '69',
    class: 'TYCS'
  },
  {
    _id: ObjectId("63f1b59cc286711c3279d274"),
    Dept: 'cs',
    Name: 'Taslim',
    Roll_no: '18',
    class: 'TYCS'
  }
]
```

```
SIWS> db.DS.drop()
true
SIWS> db.dropDatabase()
{ ok: 1, dropped: 'SIWS' }
SIWS> show dbs
admin     40.00 KiB
config   108.00 KiB
local     40.00 KiB
taslim     8.00 KiB
```

PRACTICAL NO:- 2

Data curation and management for Large-scale Data system

AIM:- Practical of Data collection, Data curation and management for Large-scale Data system (such as MongoDB).

DESCRIPTION:- Big data management refers to the organisation, administration and governance of large volumes of unstructured and structured data. A high level of data quality and accessibility for business intelligence and big data analytics applications is the aim of big data management.

PROGRAM:-

To create a database in MongoDB

Use: DATABASE_NAME

If you want to check your databases list

Use: show dbs.

create a collection using MongoDB.

Use: db.createCollection(name, options)

To Insert document in MongoDB collection.

Use: db.COLLECTION_NAME.insert(document)

To query document from MongoDB collection.

Use: db.COLLECTION_NAME.find()

To update document

Use: db.COLLECTION_NAME.insert({name},{$set:{new naem}})
To delete collection: To drop Collection
Use: db.COLLECTION_NAME.remove()
To drop DB
Use: db.COLLECTION_NAME.drop()

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
> iris
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1           5.1         3.5          1.4         0.2     setosa
2           4.9         3.0          1.4         0.2     setosa
3           4.7         3.2          1.3         0.2     setosa
4           4.6         3.1          1.5         0.2     setosa
5           5.0         3.6          1.4         0.2     setosa
6           5.4         3.9          1.7         0.4     setosa
7           4.6         3.4          1.4         0.3     setosa
8           5.0         3.4          1.5         0.2     setosa
9           4.4         2.9          1.4         0.2     setosa
10          4.9         3.1          1.5         0.1     setosa
11          5.4         3.7          1.5         0.2     setosa
12          4.8         3.4          1.6         0.2     setosa
13          4.8         3.0          1.4         0.1     setosa
14          4.3         3.0          1.1         0.1     setosa
15          5.8         4.0          1.2         0.2     setosa
16          5.7         4.4          1.5         0.4     setosa
17          5.4         3.9          1.3         0.4     setosa
18          5.1         3.5          1.4         0.3     setosa
19          5.7         3.8          1.7         0.3     setosa
20          5.1         3.8          1.5         0.3     setosa
21          5.4         3.4          1.7         0.2     setosa
22          5.1         3.7          1.5         0.4     setosa
23          4.6         3.6          1.0         0.2     setosa
24          5.1         3.3          1.7         0.5     setosa
25          4.8         3.4          1.9         0.2     setosa
26          5.0         3.0          1.6         0.2     setosa
27          5.0         3.4          1.6         0.4     setosa
28          5.2         3.5          1.5         0.2     setosa
29          5.2         3.4          1.4         0.2     setosa
30          4.7         3.2          1.6         0.2     setosa
31          4.8         3.1          1.6         0.2     setosa
32          5.4         3.4          1.5         0.4     setosa
33          5.2         4.1          1.5         0.1     setosa
34          5.5         4.2          1.4         0.2     setosa
35          4.9         3.1          1.5         0.2     setosa
```

2.

```
> data_iris<-iris[1:4]
> data_iris
   Sepal.Length Sepal.Width Petal.Length Petal.Width
1           5.1         3.5          1.4         0.2
2           4.9         3.0          1.4         0.2
3           4.7         3.2          1.3         0.2
4           4.6         3.1          1.5         0.2
5           5.0         3.6          1.4         0.2
6           5.4         3.9          1.7         0.4
7           4.6         3.4          1.4         0.3
8           5.0         3.4          1.5         0.2
9           4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
11          5.4         3.7          1.5         0.2
12          4.8         3.4          1.6         0.2
13          4.8         3.0          1.4         0.1
14          4.3         3.0          1.1         0.1
15          5.8         4.0          1.2         0.2
16          5.7         4.4          1.5         0.4
17          5.4         3.9          1.3         0.4
18          5.1         3.5          1.4         0.3
19          5.7         3.8          1.7         0.3
20          5.1         3.8          1.5         0.3
21          5.4         3.4          1.7         0.2
22          5.1         3.7          1.5         0.4
23          4.6         3.6          1.0         0.2
24          5.1         3.3          1.7         0.5
25          4.8         3.4          1.9         0.2
26          5.0         3.0          1.6         0.2
27          5.0         3.4          1.6         0.4
28          5.2         3.5          1.5         0.2
29          5.2         3.4          1.4         0.2
30          4.7         3.2          1.6         0.2
31          4.8         3.1          1.6         0.2
32          5.4         3.4          1.5         0.4
33          5.2         4.1          1.5         0.1
34          5.5         4.2          1.4         0.2
35          4.9         3.1          1.5         0.2
```

3.

```
> Cov_data<-cov(data_iris)
> Cov_data
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

PRACTICAL NO:- 3

Principal Component Analysis

AIM:- Practical of Principal Component Analysis(PCA).

DESCRIPTION:- Principal component analysis is used to extract the important information from a multivariate data table and to express this information as a set of few new variables called principal components. These new variables correspond to a linear combination of the originals.

PROGRAM:-

1. **Iris**

2. **data_iris<-iris[1:4]**
   data_iris

3. **Cov_data<-cov(data_iris)**
   Cov_data

4.

```
> Eigen_data<-eigen(Cov_data)
> Eigen_data
eigen() decomposition
$values
[1] 4.22824171 0.24267075 0.07820950 0.02383509

$vectors
            [,1]        [,2]        [,3]       [,4]
[1,]  0.36138659 -0.65658877 -0.58202985  0.3154872
[2,] -0.08452251 -0.73016143  0.59791083 -0.3197231
[3,]  0.85667061  0.17337266  0.07623608 -0.4798390
[4,]  0.35828920  0.07548102  0.54583143  0.7536574
```

5.

```
> PCA_data<-princomp(data_iris,cor="False")
> PCA_data
Call:
princomp(x = data_iris, cor = "False")

Standard deviations:
   Comp.1    Comp.2    Comp.3    Comp.4 
2.0494032 0.4909714 0.2787259 0.1538707 

 4  variables and  150 observations.
```

6.

```
> Eigen_data$values
[1] 4.22824171 0.24267075 0.07820950 0.02383509
> PCA_data$sdev^2
    Comp.1     Comp.2     Comp.3     Comp.4 
4.20005343 0.24105294 0.07768810 0.02367619 
```

7.

```
> PCA_data$sdev^2
    Comp.1     Comp.2     Comp.3     Comp.4 
4.20005343 0.24105294 0.07768810 0.02367619 
```

8.

```
> PCA_data$loadings[,1:4]
                  Comp.1      Comp.2      Comp.3     Comp.4
Sepal.Length  0.36138659  0.65658877  0.58202985  0.3154872
Sepal.Width  -0.08452251  0.73016143 -0.59791083 -0.3197231
Petal.Length  0.85667061 -0.17337266 -0.07623608 -0.4798390
Petal.Width   0.35828920 -0.07548102 -0.54583143  0.7536574
```

9.

```
> Eigen_data$vectors
            [,1]        [,2]        [,3]       [,4]
[1,]  0.36138659 -0.65658877 -0.58202985  0.3154872
[2,] -0.08452251 -0.73016143  0.59791083 -0.3197231
[3,]  0.85667061  0.17337266  0.07623608 -0.4798390
[4,]  0.35828920  0.07548102  0.54583143  0.7536574
```

4. **Eigen_data<-eigen(Cov_data)**
   Eigen_data

5. **PCA_data<-princomp(data_iris,cor="False")**
   PCA_data

6. **Eigen_data$values**

7. **PCA_data$sdev^2**

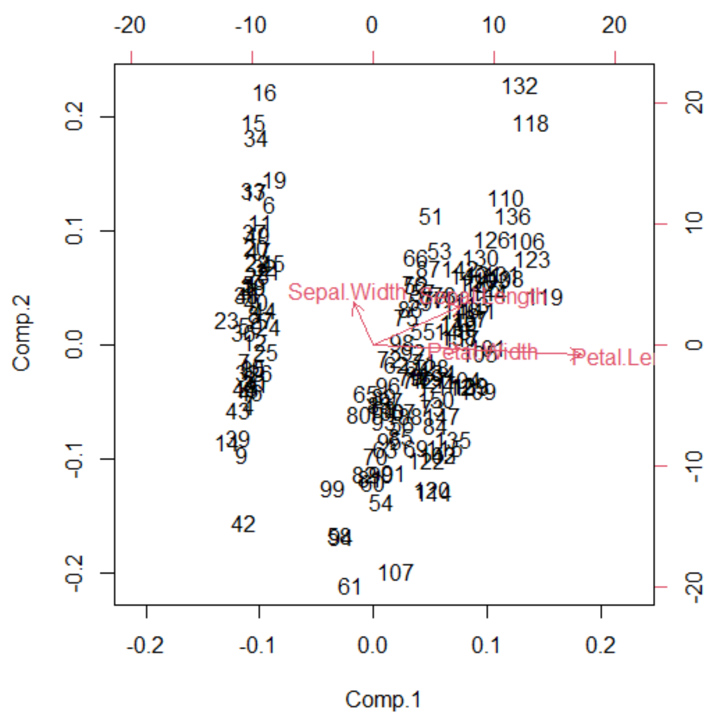
8. **PCA_data$loadings[,1:4]**

9. **Eigen_data$vectors**

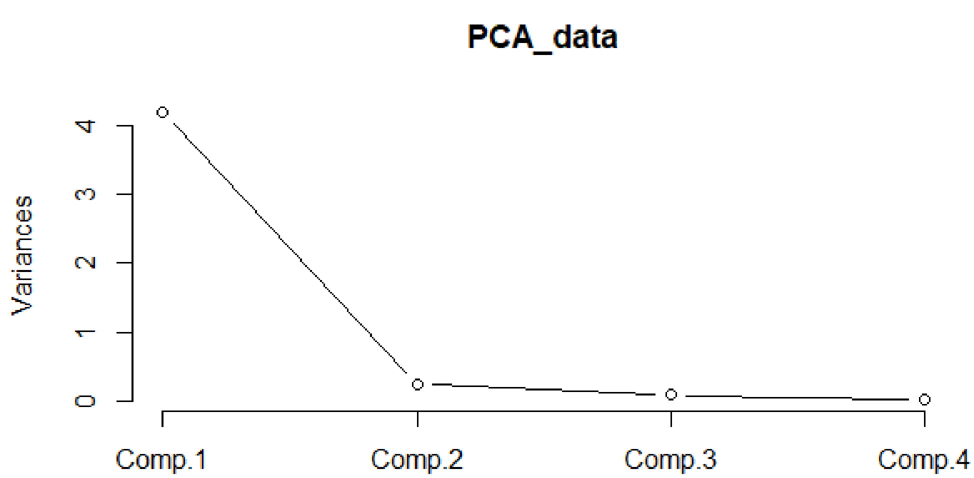
10.

```
> summary(PCA_data)
Importance of components:
                          Comp.1     Comp.2     Comp.3      Comp.4
Standard deviation     2.0494032 0.49097143 0.27872586 0.153870700
Proportion of Variance 0.9246187 0.05306648 0.01710261 0.005212184
Cumulative Proportion  0.9246187 0.97768521 0.99478782 1.000000000
```

11.



12.

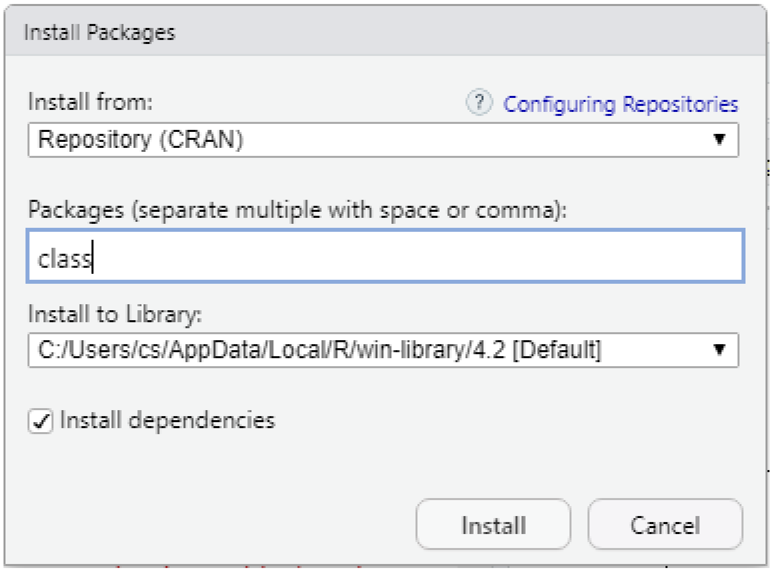


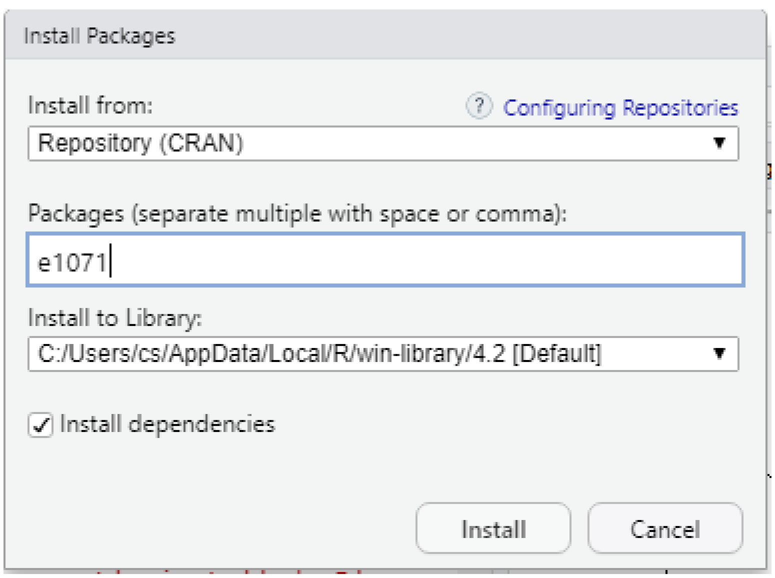
10. summary(PCA_data)

11. biplot(PCA_data)

12. screeplot(PCA_data,type = "lines")

13.





14.

```
> screeplot(PCA_data,type = "lines")
> model2=PCA_data$loadings[,1]
> model2_scores<-as.matrix(data_iris)%*%model2
```

13. Install Following 2 packages.

1) class
2) e1071

14.screeplot(PCA_data,type = "lines")
model2=PCA_data$loadings[,1]
model2_scores<-as.matrix(data_iris)%*%model2

15.

```
> mod1<-naiveBayes(iris[,1:4],iris[,5])
> table(predict(mod1,iris[,1:4]),iris[,5])

             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          3        47
```

16.

```
> mod2<-naiveBayes(model2_scores,iris[,5])
> table(predict(mod2,model2_scores),iris[,5])

             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         46         5
  virginica       0          4        45
```

15. **mod1<-naiveBayes(iris[,1:4],iris[,5])**
    table(predict(mod1,iris[,1:4]),iris[,5]

16. **mod2<-naiveBayes(model2_scores,iris[,5])**
    table(predict(mod2,model2_scores),iris[,5])

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
Console   Terminal ×   Jobs ×
R 4.2.2 · ~/Taslim/
> iris
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1           5.1         3.5          1.4         0.2     setosa
2           4.9         3.0          1.4         0.2     setosa
3           4.7         3.2          1.3         0.2     setosa
4           4.6         3.1          1.5         0.2     setosa
5           5.0         3.6          1.4         0.2     setosa
6           5.4         3.9          1.7         0.4     setosa
7           4.6         3.4          1.4         0.3     setosa
8           5.0         3.4          1.5         0.2     setosa
9           4.4         2.9          1.4         0.2     setosa
10          4.9         3.1          1.5         0.1     setosa
11          5.4         3.7          1.5         0.2     setosa
12          4.8         3.4          1.6         0.2     setosa
13          4.8         3.0          1.4         0.1     setosa
14          4.3         3.0          1.1         0.1     setosa
15          5.8         4.0          1.2         0.2     setosa
16          5.7         4.4          1.5         0.4     setosa
17          5.4         3.9          1.3         0.4     setosa
18          5.1         3.5          1.4         0.3     setosa
19          5.7         3.8          1.7         0.3     setosa
20          5.1         3.8          1.5         0.3     setosa
21          5.4         3.4          1.7         0.2     setosa
22          5.1         3.7          1.5         0.4     setosa
23          4.6         3.6          1.0         0.2     setosa
24          5.1         3.3          1.7         0.5     setosa
25          4.8         3.4          1.9         0.2     setosa
26          5.0         3.0          1.6         0.2     setosa
27          5.0         3.4          1.6         0.4     setosa
28          5.2         3.5          1.5         0.2     setosa
29          5.2         3.4          1.4         0.2     setosa
30          4.7         3.2          1.6         0.2     setosa
31          4.8         3.1          1.6         0.2     setosa
32          5.4         3.4          1.5         0.4     setosa
33          5.2         4.1          1.5         0.1     setosa
34          5.5         4.2          1.4         0.2     setosa
35          4.9         3.1          1.5         0.2     setosa
36          5.0         3.2          1.2         0.2     setosa
37          5.5         3.5          1.3         0.2     setosa
```

2.

```
Console   Terminal ×   Jobs ×
R 4.2.2 · ~/Taslim/
150          5.9         3.0          5.1         1.8  virginica
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

PRACTICAL NO:- 4

Data Clustering Using Clustering Algorithm

AIM:- Perform the data clustering using clustering algorithm.

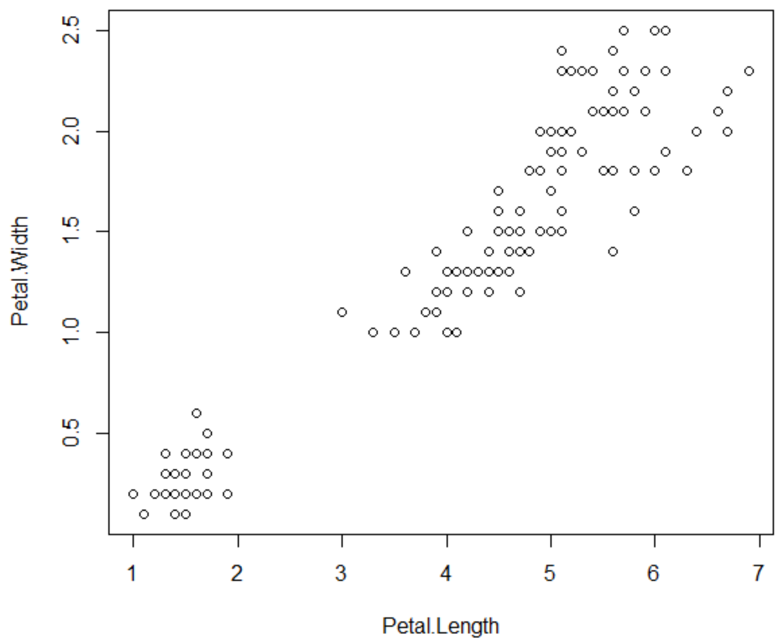
DESCRIPTION:- Data clustering is the most commonly used clustering algorithm. It's a centroid-based algorithm and the simplest unsupervised learning algorithm. This algorithm tries to minimize the variance of data points within a cluster.

PROGRAM:-

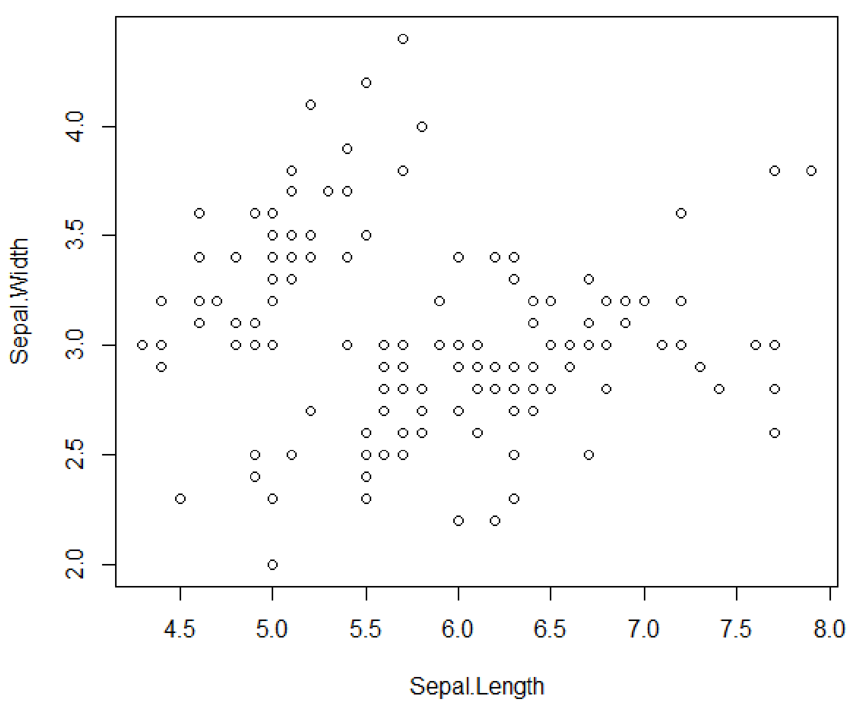
1. **Iris**

2. **summary(iris)**

3.


Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

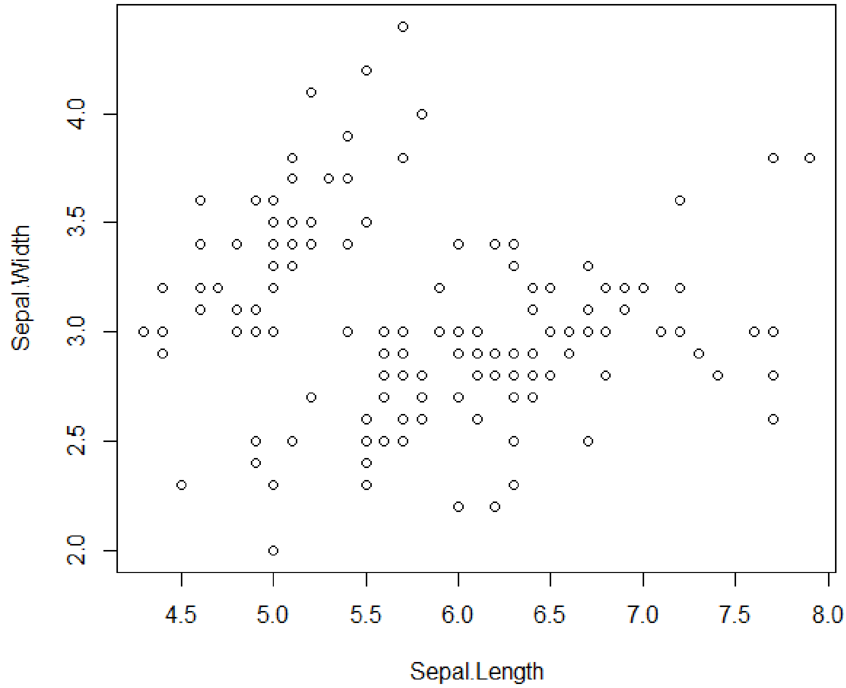
4.


Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

3. **plot(iris [c("Petal.Length","Petal.Width")])**

4. **plot(iris [c("Sepal.Length","Sepal.Width")])**

5.



6.

```
Console  Terminal ×  Jobs ×
R 4.2.2 · ~/Taslim/
> newiris$Species <- NULL
> newiris
   Sepal.Length Sepal.Width Petal.Length Petal.Width
1           5.1         3.5          1.4         0.2
2           4.9         3.0          1.4         0.2
3           4.7         3.2          1.3         0.2
4           4.6         3.1          1.5         0.2
5           5.0         3.6          1.4         0.2
6           5.4         3.9          1.7         0.4
7           4.6         3.4          1.4         0.3
8           5.0         3.4          1.5         0.2
9           4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
11          5.4         3.7          1.5         0.2
12          4.8         3.4          1.6         0.2
13          4.8         3.0          1.4         0.1
14          4.3         3.0          1.1         0.1
15          5.8         4.0          1.2         0.2
16          5.7         4.4          1.5         0.4
17          5.4         3.9          1.3         0.4
18          5.1         3.5          1.4         0.3
19          5.7         3.8          1.7         0.3
20          5.1         3.8          1.5         0.3
21          5.4         3.4          1.7         0.2
22          5.1         3.7          1.5         0.4
23          4.6         3.6          1.0         0.2
24          5.1         3.3          1.7         0.5
25          4.8         3.4          1.9         0.2
26          5.0         3.0          1.6         0.2
27          5.0         3.4          1.6         0.4
28          5.2         3.5          1.5         0.2
29          5.2         3.4          1.4         0.2
30          4.7         3.2          1.6         0.2
31          4.8         3.1          1.6         0.2
32          5.4         3.4          1.5         0.4
33          5.2         4.1          1.5         0.1
34          5.5         4.2          1.4         0.2
35          4.9         3.1          1.5         0.2
36          5.0         3.2          1.2         0.2
37          5.5         3.5          1.3         0.2
```

5. **newiris <- iris**

6. **newiris$Species <- NULL**
   newiris

7.

```
> (kc <- kmeans(newiris,3))
K-means clustering with 3 clusters of sizes 62, 50, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     5.006000    3.428000     1.462000    0.246000
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [40] 2 2 2 2 2 2 2 2 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3
 [79] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 3 3 3 1 3 3 3 3 3 3 1 1 3 3
[118] 3 3 1 3 1 3 1 3 3 1 1 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3 3 1

Within cluster sum of squares by cluster:
[1] 39.82097 15.15100 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

8.

```
> kc$size
[1] 62 50 38
```

9.

```
> kc$cluster
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [40] 2 2 2 2 2 2 2 2 2 2 2 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3
 [79] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 3 3 3 1 3 3 3 3 3 3 1 1 3 3
[118] 3 3 1 3 1 3 1 3 3 1 1 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3 3 1
```

10.

```
> kc$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     5.006000    3.428000     1.462000    0.246000
3     6.850000    3.073684     5.742105    2.071053
```

11.

```
> kc$withinss
[1] 39.82097 15.15100 23.87947
```

12.

```
> kc$betweenss
[1] 602.5192
```

13.

```
> table(iris$Species,kc$cluster)

              1  2  3
  setosa      0 50  0
  versicolor 48  0  2
  virginica  14  0 36
```

7. **(kc <- kmeans(newiris,3))**

8. **kc$size**

9. **kc$cluster**

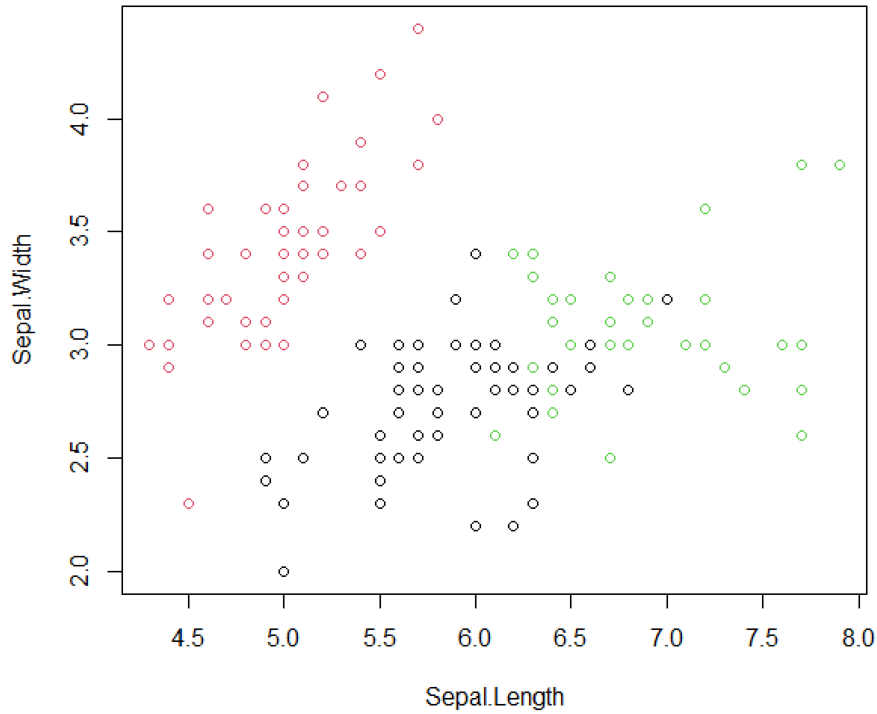
10. **kc$centers**

11. **kc$withinss**

12. **kc$betweenss**

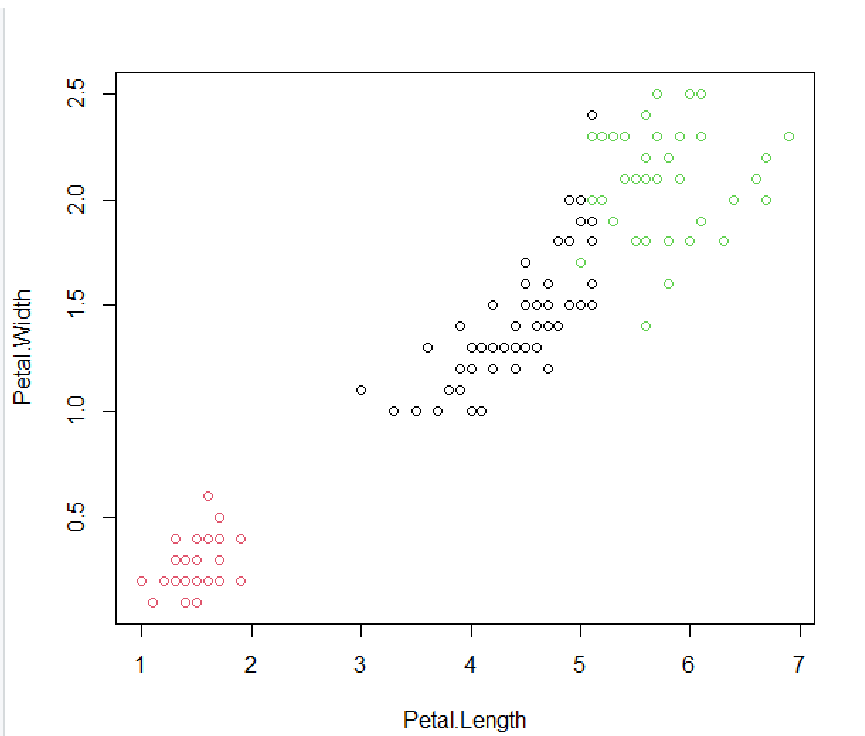
13. **table(iris$Species,kc$cluster)**

14.



Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

15.

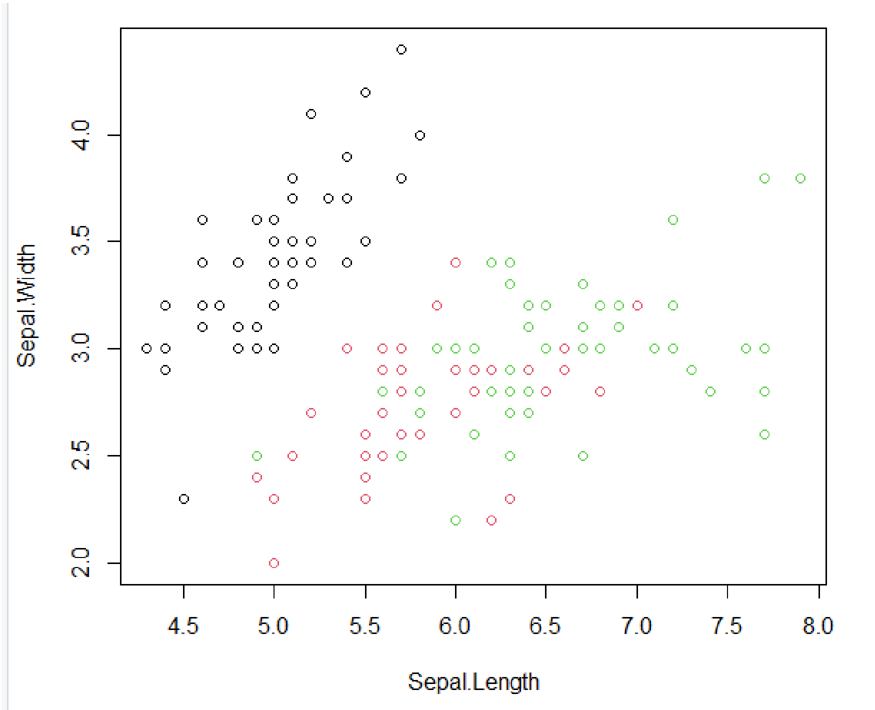


Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

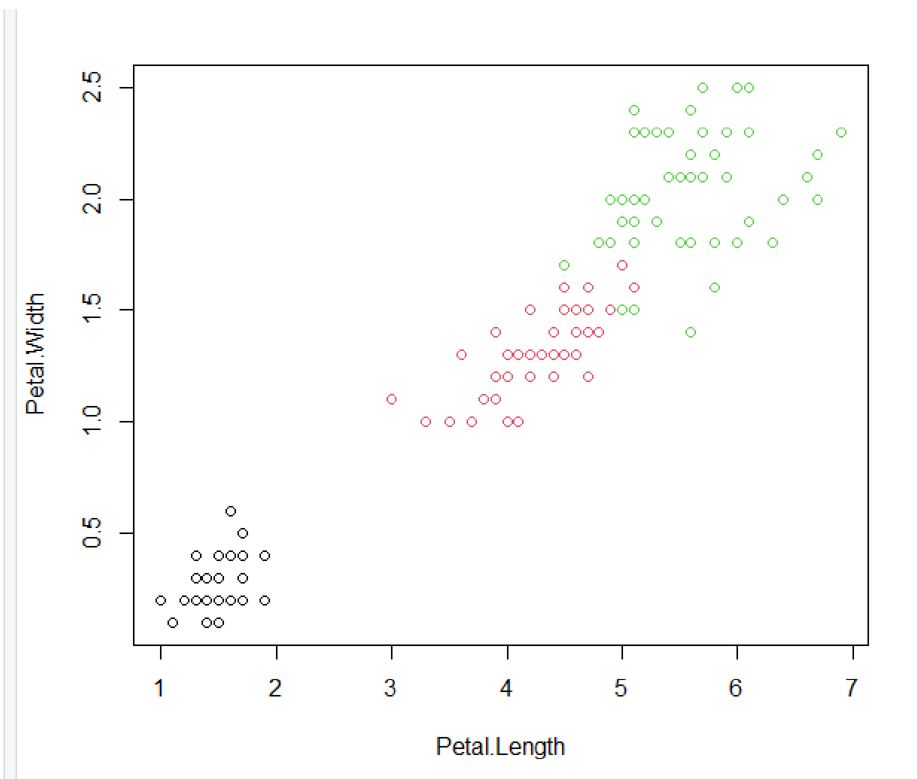
14.plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)

15.plot(newiris[c("Petal.Length","Petal.Width")],col=kc$cluster)

16.


Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

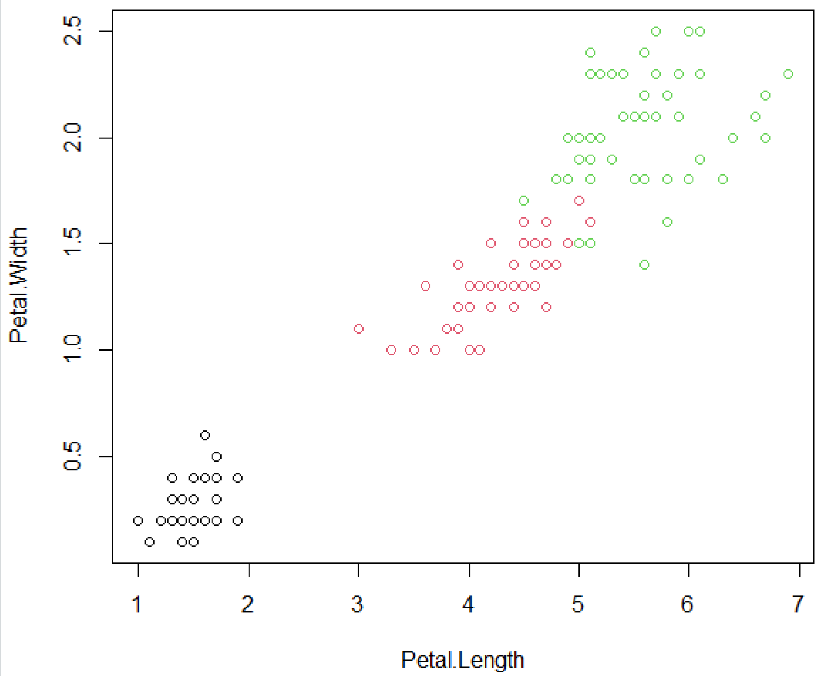
17.


Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

16.plot(newiris[c("Sepal.Length","Sepal.Width")],col=iris$Species)

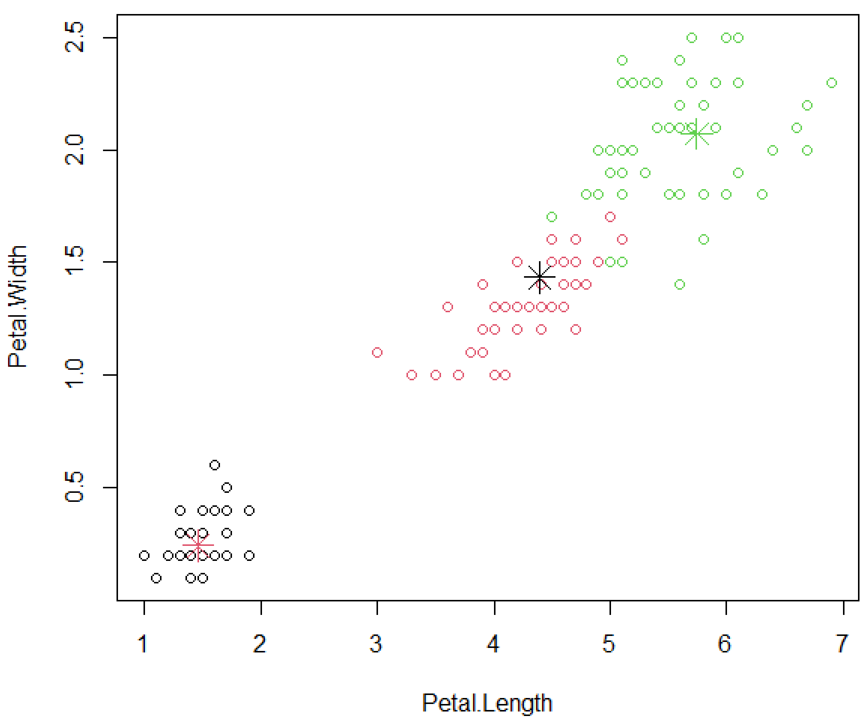
17.plot(newiris[c("Petal.Length","Petal.Width")],col=iris$Species)

18.



Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

19.



Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

18. **points(kc$centers[,c("Sepal.Length","Sepal.Width")],col=1:3,pch=8,cex=2)**

19. **points(kc$centers[,c("Petal.Length","Petal.Width")],col=1:3,pch=8,cex=2)**

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
> data("AirPassengers")
> class(AirPassengers)
[1] "ts"
```

2.

```
> start(AirPassengers)
[1] 1949    1
```

3.

```
> end(AirPassengers)
[1] 1960   12
```

4.

```
> frequency(AirPassengers)
[1] 12
```

5.

```
> summary(AirPassengers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  104.0   180.0   265.5   280.3   360.5   622.0 
```

PRACTICAL NO:- 5

Time – Series Forecasting

AIM:- Practical of Time – series forecasting.

DESCRIPTION:- Time series data analysis is increasingly important due to the massive production of such data through the internet of things, the digitalization of healthcare, and the rise of smart cities.

PROGRAM:-

1. **data("AirPassengers")**
   class(AirPassengers)

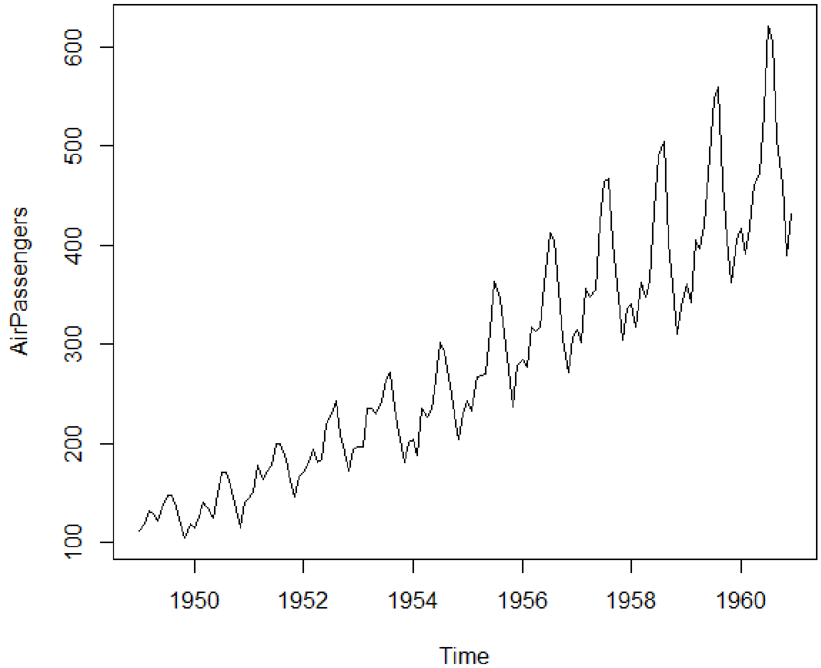
2. **start(AirPassengers)**

3. **end(AirPassengers)**

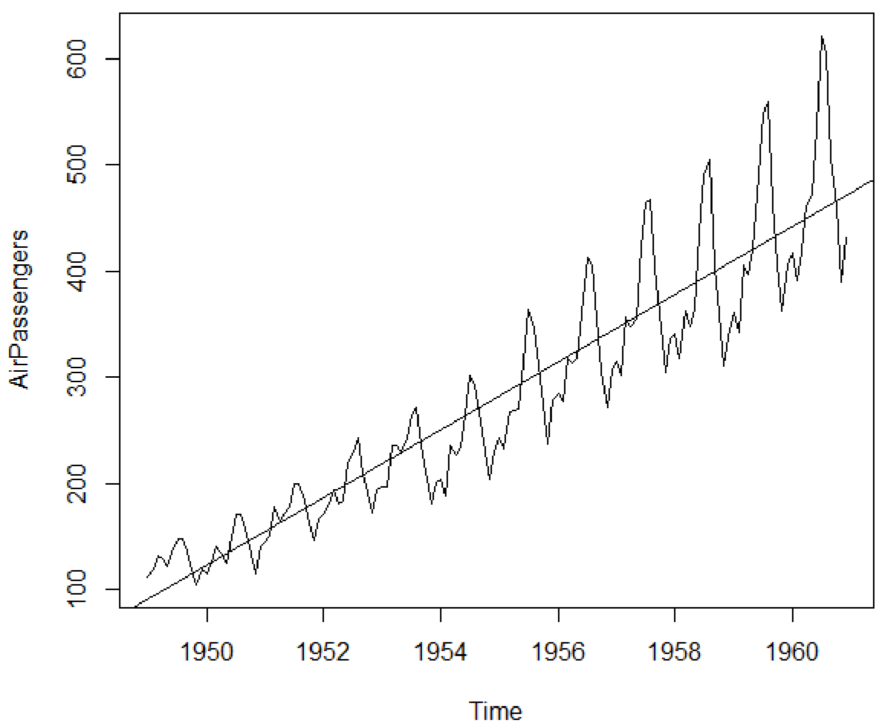
4. **frequency(AirPassengers)**

5. **summary(AirPassengers)**

6.


AirPassengers
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960
Time

7.


AirPassengers
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960
Time

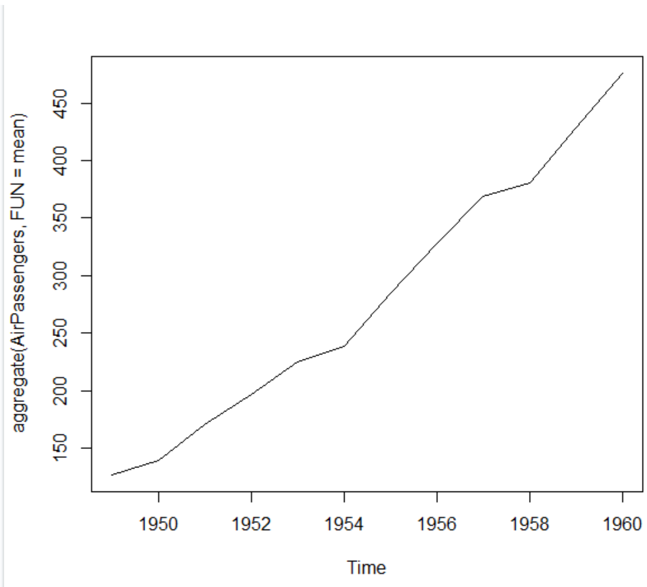
6. **plot(AirPassengers)**

7. **abline(reg=lm(AirPassengers~time(AirPassengers)))**

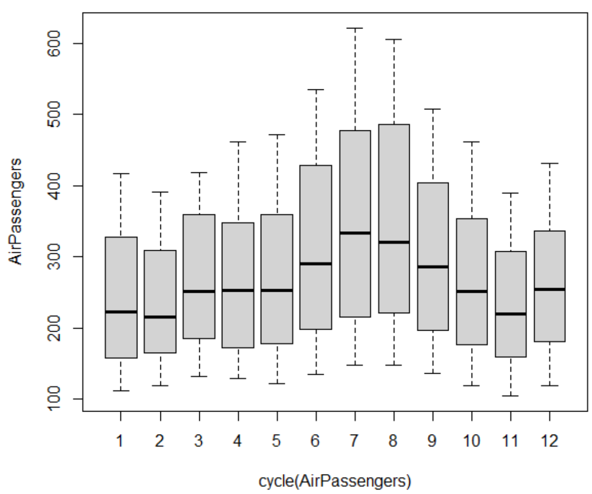
8.

```
> cycle(AirPassengers)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949   1   2   3   4   5   6   7   8   9  10  11  12
1950   1   2   3   4   5   6   7   8   9  10  11  12
1951   1   2   3   4   5   6   7   8   9  10  11  12
1952   1   2   3   4   5   6   7   8   9  10  11  12
1953   1   2   3   4   5   6   7   8   9  10  11  12
1954   1   2   3   4   5   6   7   8   9  10  11  12
1955   1   2   3   4   5   6   7   8   9  10  11  12
1956   1   2   3   4   5   6   7   8   9  10  11  12
1957   1   2   3   4   5   6   7   8   9  10  11  12
1958   1   2   3   4   5   6   7   8   9  10  11  12
1959   1   2   3   4   5   6   7   8   9  10  11  12
1960   1   2   3   4   5   6   7   8   9  10  11  12
```

9.



10.

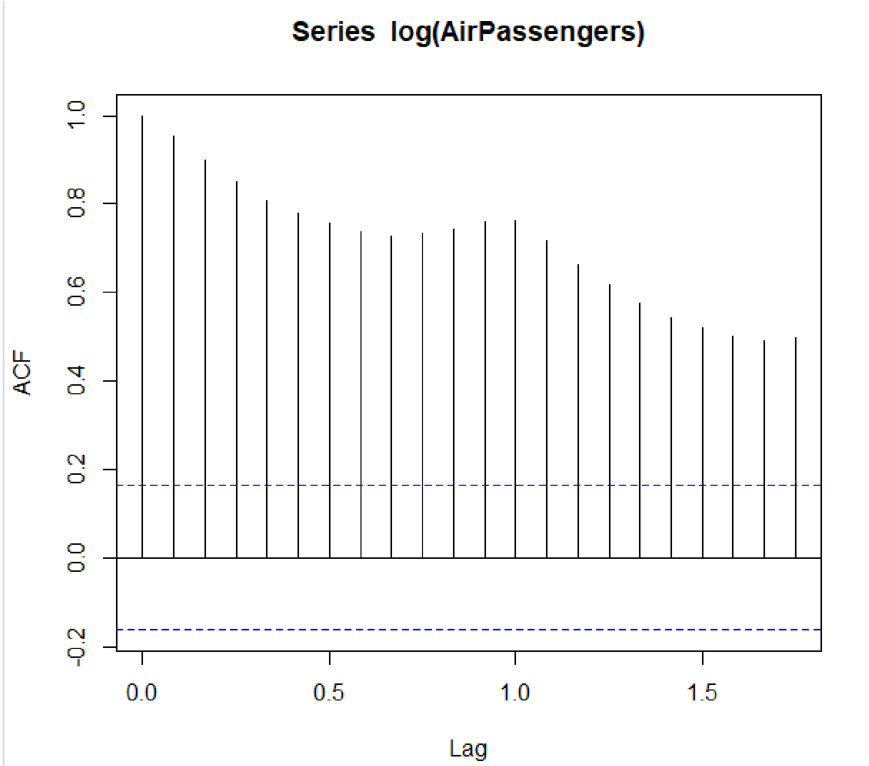


8. cycle(AirPassengers)

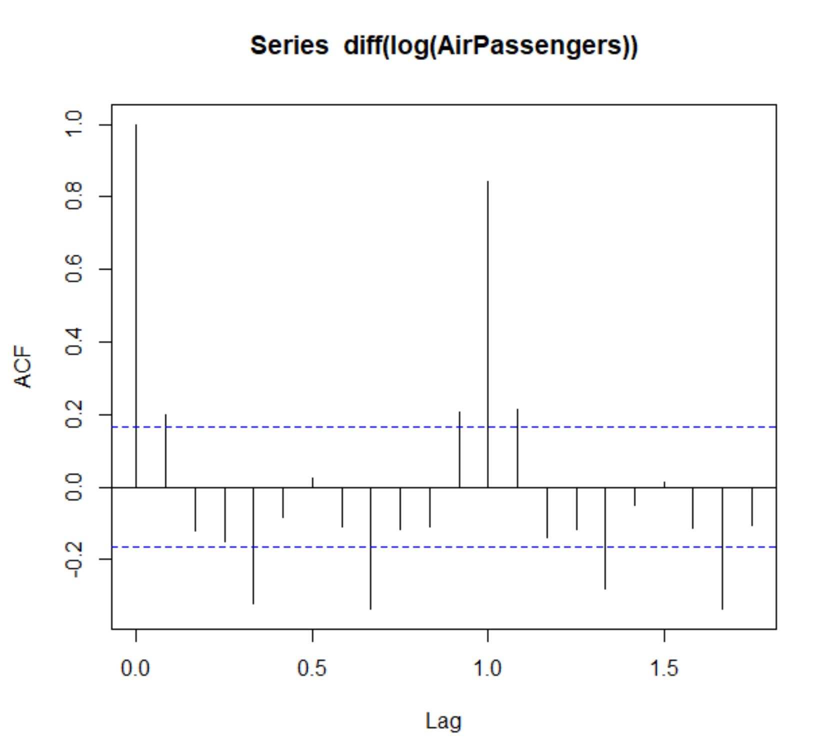
9. plot(aggregate(AirPassengers,FUN=mean))

10.boxplot(AirPassengers~cycle(AirPassengers))

11.


Series log(AirPassengers)
ACF
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
0.0
0.5
1.0
1.5
Lag

12.


Series diff(log(AirPassengers))
ACF
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
0.0
0.5
1.0
1.5
Lag

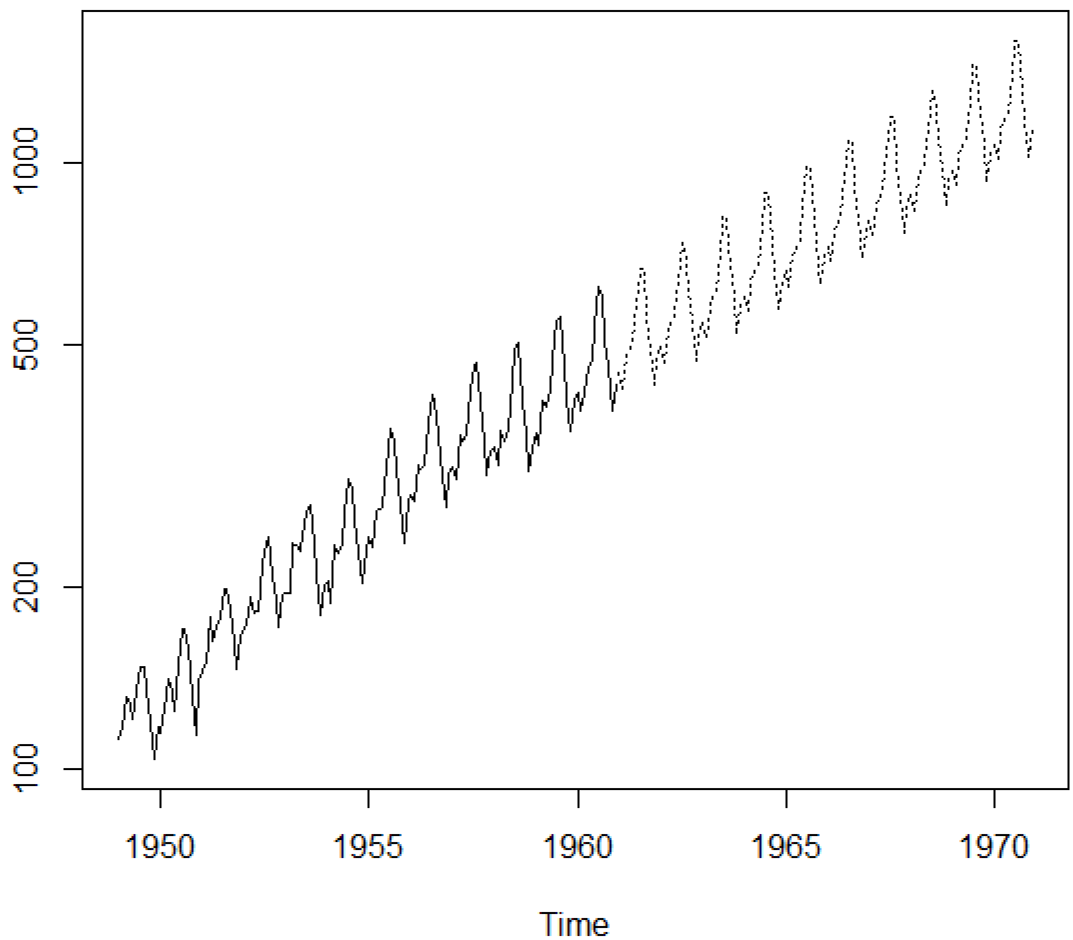
11. acf(log(AirPassengers))

12.acf(diff(log(AirPassengers)))

13.

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

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
```

14.



13. (fit<-arima(log(AirPassengers),c(0,1,1),seasonal=list(order=c(0,1,1),period=12)))
pred<-predict(fit,n.ahead = 10*12)

14. ts.plot(AirPassengers,2.718^pred$pred,log="y",lty=c(1,3))

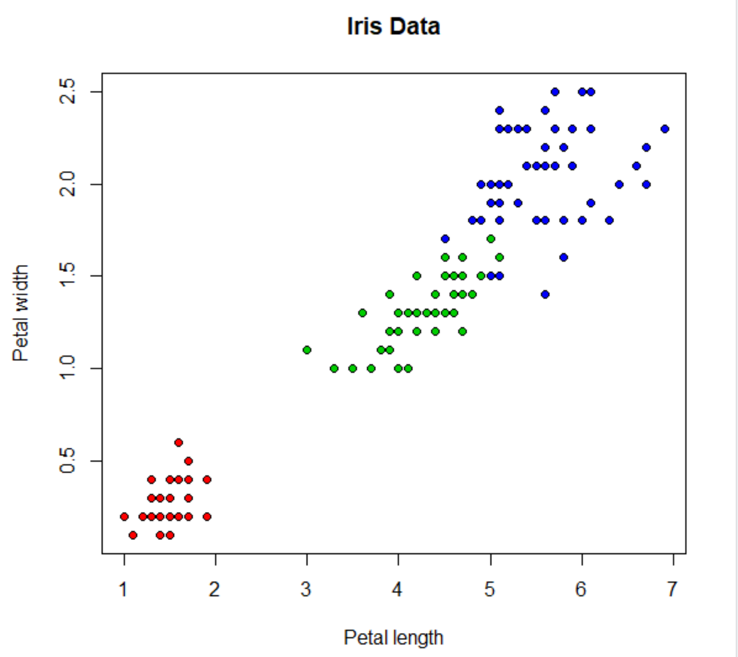
CONCLUSION:- The above program has been executed successfully.

Output:-

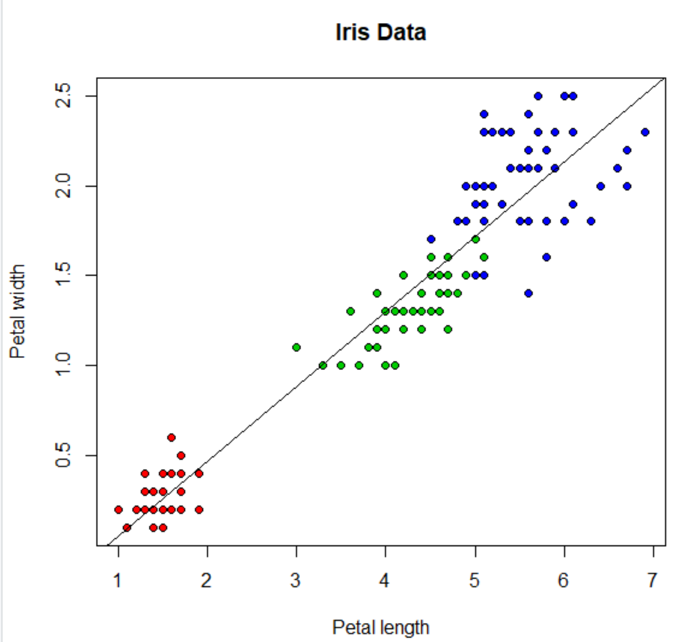
1.

```
> lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients
 Intercept          X 
-0.3630755  0.4157554 
```

2.



3



PRACTICAL NO:- 6

Simple And Multiple Linear Regression

AIM:- Practical of Simple and Multiple Linear Regression.

DESCRIPTION:- What's the difference between simple and multiple linear regression techniques? As noted above, the simple linear method measures one independent variable against one dependent variable. The multiple linear technique is used when there are at least two independent variables against the one dependent variable.

PROGRAM:-

1. **lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients**

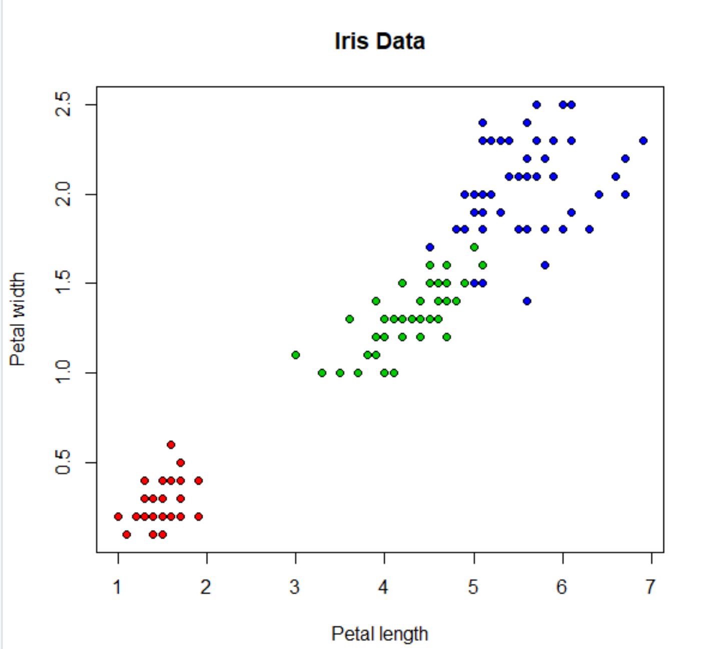
2. **plot(iris$Petal.Length,iris$Petal.Width,pch=21,bg=c("red","green3","blue")[unclass(iris$Species)],main="IrisData",xlab="Petal length",ylab="Petal width")**

3. **abline(lsfit(iris$Petal.Length,iris$Petal.Width)$coefficients, col="black")**

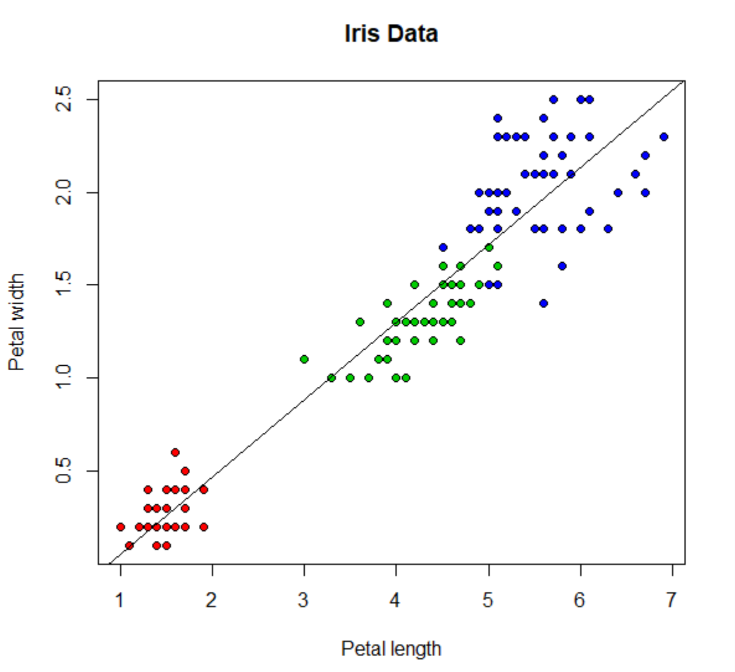
4.

```
> lm(Petal.Width ~ Petal.Length, data=iris)$coefficients
 (Intercept) Petal.Length
  -0.3630755    0.4157554
```

5.



6.



4. **lm(Petal.Width ~ Petal.Length, data=iris)$coefficients**

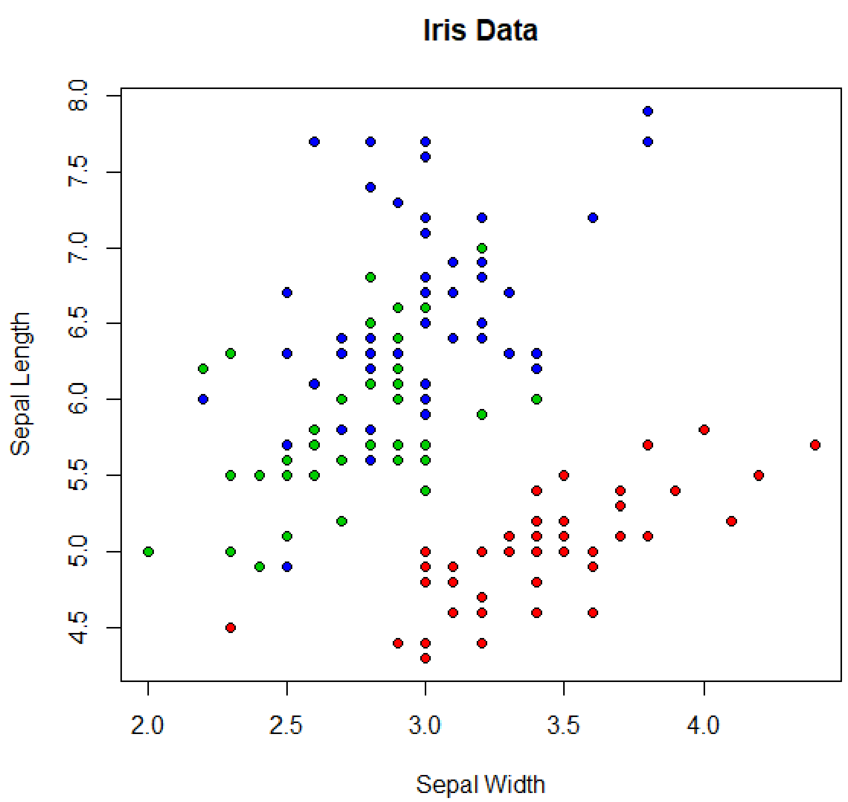
5. **plot(iris$Petal.Length, iris$Petal.Width, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length", ylab="Petal width")**

6. **abline(lm(Petal.Width ~ Petal.Length, data=iris)$coefficients, col="black")**

7.

```
> summary(lm(Petal.Width ~ Petal.Length, data=iris))

Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.56515 -0.12358 -0.01898  0.13288  0.64272 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -0.363076   0.039762  -9.131  4.7e-16 ***
Petal.Length  0.415755   0.009582  43.387  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2065 on 148 degrees of freedom
Multiple R-squared:  0.9271,	Adjusted R-squared:  0.9266 
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
```

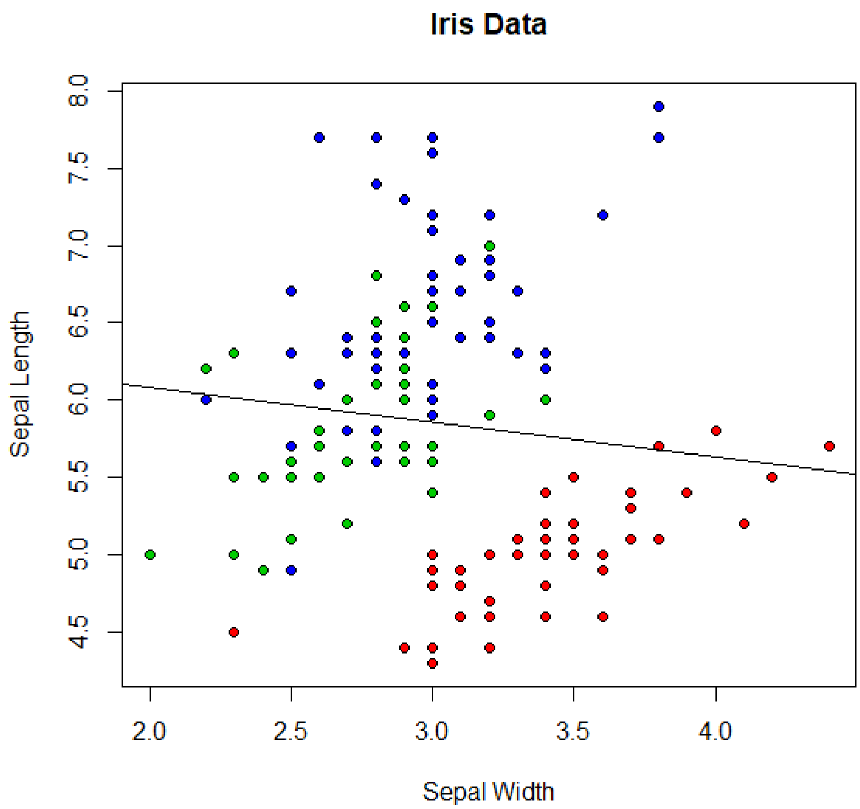
8.



7. **summary(lm(Petal.Width ~ Petal.Length, data=iris))**

8. **plot(iris$Sepal.Width, iris$Sepal.Length, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Sepal Width", ylab="Sepal Length")**

9.



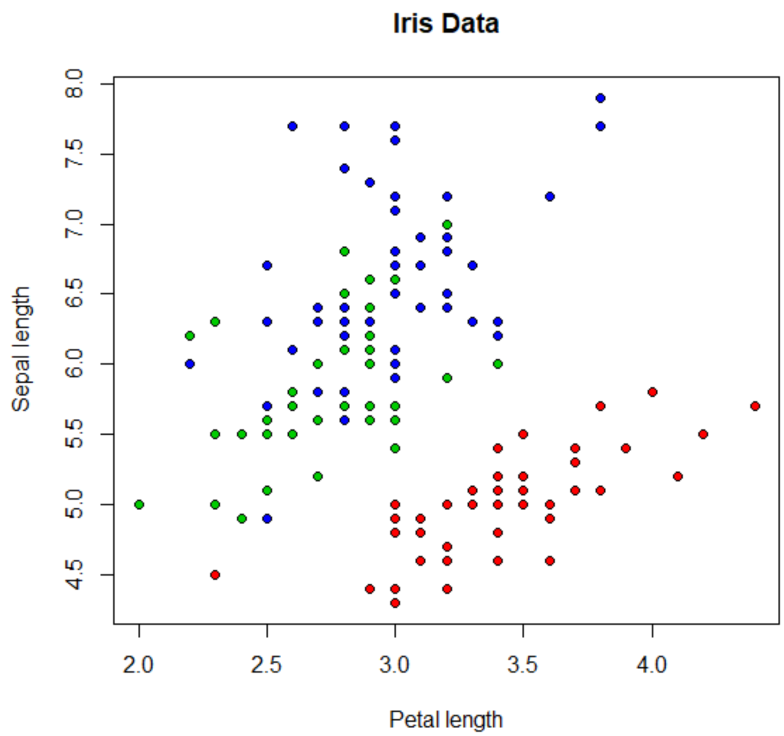
10.

```
> summary(lm(Sepal.Length ~ Sepal.Width, data=iris))

Call:
lm(formula = Sepal.Length ~ Sepal.Width, data = iris)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5561 -0.6333 -0.1120  0.5579  2.2226 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   6.5262     0.4789   13.63   <2e-16 ***
Sepal.Width  -0.2234     0.1551   -1.44    0.152    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8251 on 148 degrees of freedom
Multiple R-squared:  0.01382,	Adjusted R-squared:  0.007159 
F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

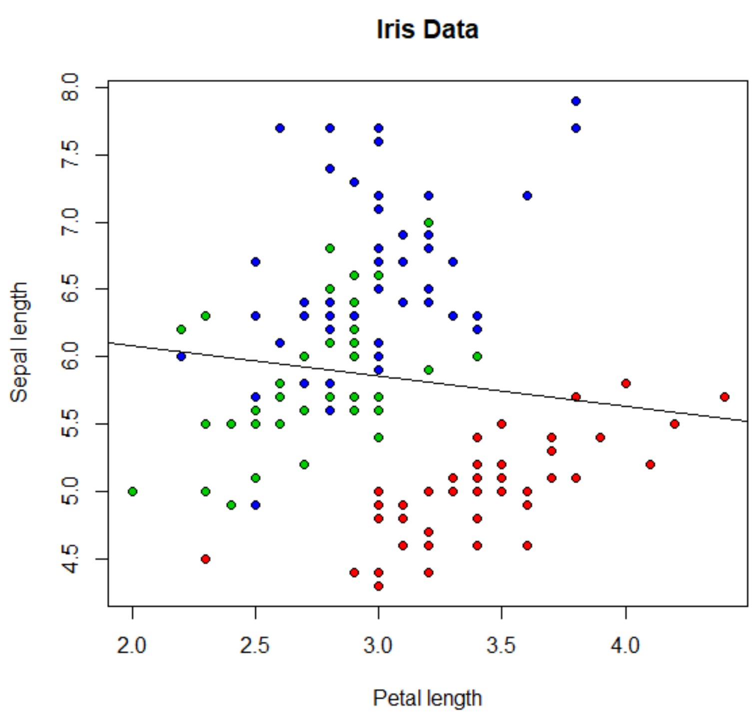
9. **abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")**

10. **summary(lm(Sepal.Length ~ Sepal.Width, data=iris))**

11.


Iris Data
Sepal length
Petal length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

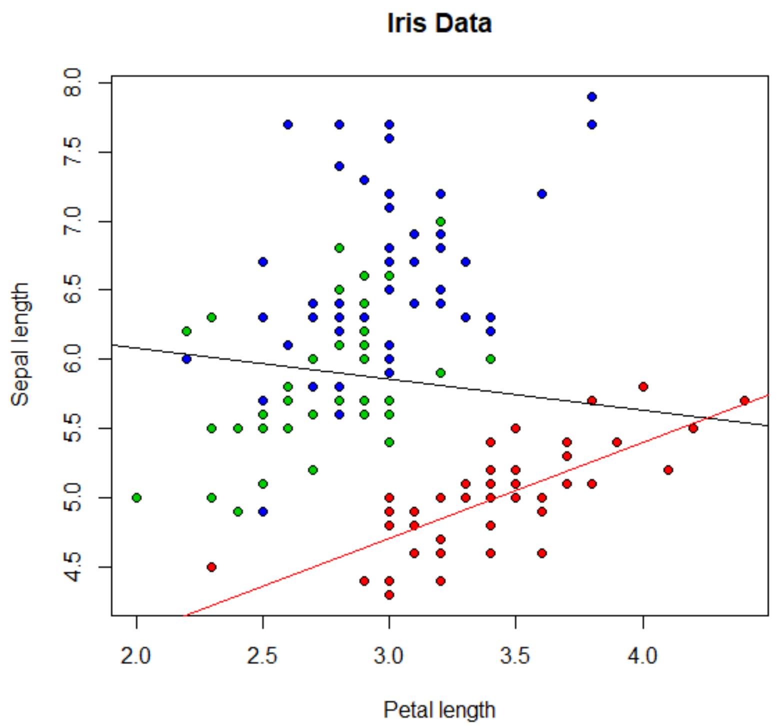
12.


Iris Data
Sepal length
Petal length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

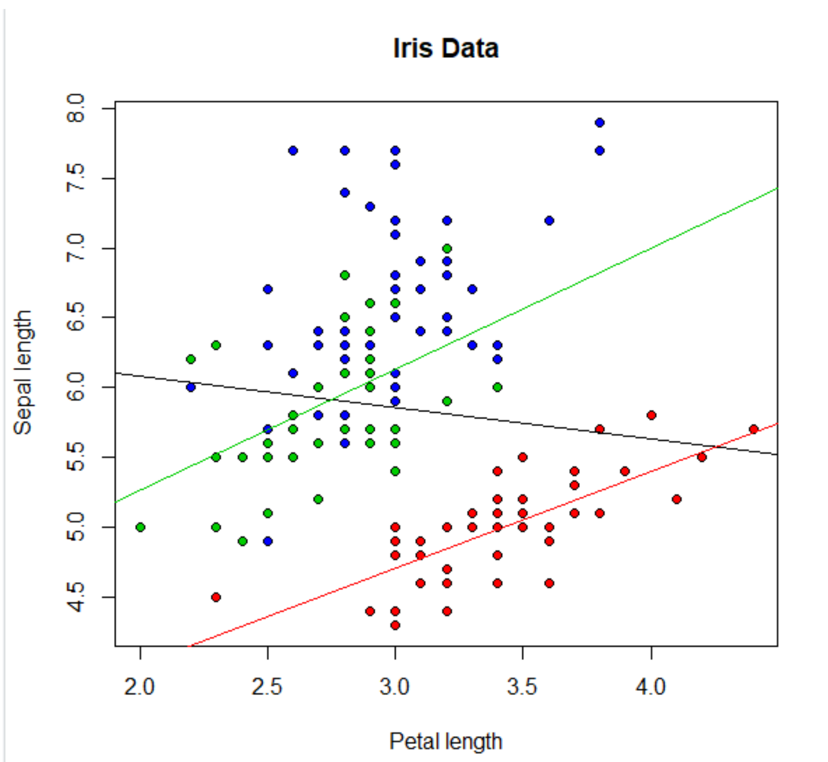
11. plot(iris$Sepal.Width, iris$Sepal.Length, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length", ylab="Sepal length")

12.abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")

13.


Iris Data
Sepal length
Petal length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

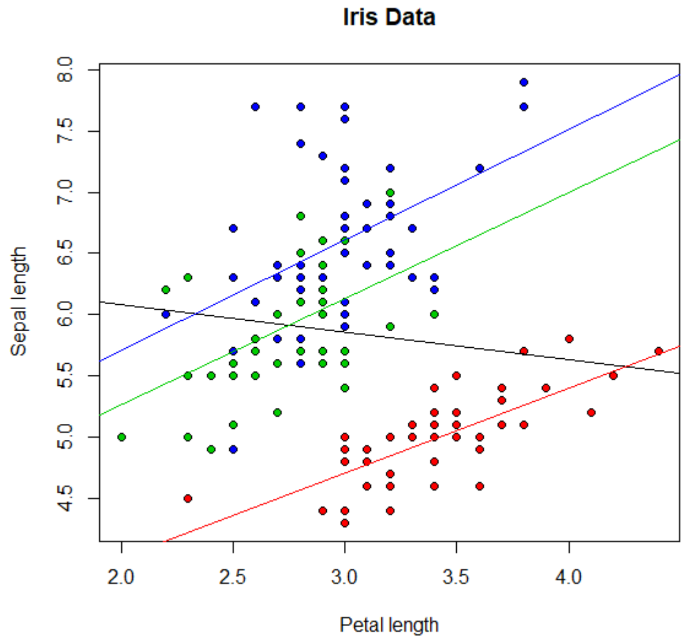
14.


Iris Data
Sepal length
Petal length
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
2.0
2.5
3.0
3.5
4.0

13. **abline(lm(Sepal.Length~Sepal.Width,data=iris[which(iris$Species=="setosa"),])$coefficients, col="red"z**

14. **abline(lm(Sepal.Length~Sepal.Width,data=iris[which(iris$Species=="versicolor"),])$coefficients, col="green3")**

15.



16.

```
> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients
(Intercept) Sepal.Width
  2.6390012   0.6904897
```

17.

```
> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficien
ts
(Intercept) Sepal.Width
  3.5397347   0.8650777
```

18.

```
> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficient
s
(Intercept) Sepal.Width
  3.9068365   0.9015345
```

19.

```
> lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients
                Speciessetosa             Speciesversicolor
                    2.6390012                     3.5397347
              Speciesvirginica      Sepal.Width:Speciessetosa
                    3.9068365                     0.6904897
Sepal.Width:Speciesversicolor  Sepal.Width:Speciesvirginica
                    0.8650777                     0.9015345
```

15. **abline(lm(Sepal.Length~Sepal.Width,data=iris[which(iris$Species=="virginica"),])$coefficients, col="blue")**

16. **lm(Sepal.Length~Sepal.Width,data=iris[which(iris$Species=="setosa"),])$coefficients**

17. **lm(Sepal.Length~Sepal.Width,data=iris[which(iris$Species=="versicolor"),])$coefficients**

18. **lm(Sepal.Length~Sepal.Width,data=iris[which(iris$Species=="virginica"),])$coefficients**

19. **lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients**

20.

```
> summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris))

Call:
lm(formula = Sepal.Length ~ Sepal.Width:Species + Species - 1,
    data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.26067 -0.25861 -0.03305  0.18929  1.44917

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
Speciessetosa                   2.6390     0.5715   4.618 8.53e-06 ***
Speciesversicolor               3.5397     0.5580   6.343 2.74e-09 ***
Speciesvirginica                3.9068     0.5827   6.705 4.25e-10 ***
Sepal.Width:Speciessetosa       0.6905     0.1657   4.166 5.31e-05 ***
Sepal.Width:Speciesversicolor   0.8651     0.2002   4.321 2.88e-05 ***
Sepal.Width:Speciesvirginica    0.9015     0.1948   4.628 8.16e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4397 on 144 degrees of freedom
Multiple R-squared:  0.9947,    Adjusted R-squared:  0.9944
F-statistic:  4478 on 6 and 144 DF,  p-value: < 2.2e-16
```

21.

```
> summary(step(lm(Sepal.Length ~ Sepal.Width * Species, data=iris)))
Start:  AIC=-240.59
Sepal.Length ~ Sepal.Width * Species

                      Df Sum of Sq    RSS     AIC
- Sepal.Width:Species  2   0.15719 28.004 -243.75
<none>                             27.846 -240.59

Step:  AIC=-243.74
Sepal.Length ~ Sepal.Width + Species

              Df Sum of Sq     RSS     AIC
<none>                      28.004 -243.75
- Sepal.Width  1    10.953  38.956 -196.23
- Species      2    72.752 100.756  -55.69

Call:
lm(formula = Sepal.Length ~ Sepal.Width + Species, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.30711 -0.25713 -0.05325  0.19542  1.41253

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.2514     0.3698   6.089 9.57e-09 ***
Sepal.Width         0.8036     0.1063   7.557 4.19e-12 ***
Speciesversicolor   1.4587     0.1121  13.012  < 2e-16 ***
Speciesvirginica    1.9468     0.1000  19.465  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.438 on 146 degrees of freedom
Multiple R-squared:  0.7259,    Adjusted R-squared:  0.7203
F-statistic: 128.9 on 3 and 146 DF,  p-value: < 2.2e-16
```

20. summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)

21.summary(step(lm(Sepal.Length ~ Sepal.Width * Species, data=iris)))

22.

```
> lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients
              Speciessetosa             Speciesversicolor 
                  2.6390012                     3.5397347 
           Speciesvirginica     Sepal.Width:Speciessetosa 
                  3.9068365                     0.6904897 
Sepal.Width:Speciesversicolor  Sepal.Width:Speciesvirginica 
                  0.8650777                     0.9015345 
```

23.

```
> lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients
                  (Intercept)             Speciesversicolor 
                    2.6390012                     0.9007335 
             Speciesvirginica     Sepal.Width:Speciessetosa 
                    1.2678352                     0.6904897 
Sepal.Width:Speciesversicolor  Sepal.Width:Speciesvirginica 
                    0.8650777                     0.9015345 
```

22. lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients

23.lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
> input <-mtcars[,c("am","cyl","hp","wt")]
>
> print(head(input))
                  am cyl  hp    wt
Mazda RX4          1   6 110 2.620
Mazda RX4 Wag      1   6 110 2.875
Datsun 710         1   4  93 2.320
Hornet 4 Drive     0   6 110 3.215
Hornet Sportabout  0   8 175 3.440
Valiant            0   6 105 3.460
```

2.

```
> input <-mtcars[,c("am","cyl","hp","wt")]
>
> am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)
>
> print(summary(am.data))

Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.17272  -0.14907  -0.01464   0.14116   1.27641

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 19.70288    8.11637   2.428   0.0152 *
cyl          0.48760    1.07162   0.455   0.6491
hp           0.03259    0.01886   1.728   0.0840 .
wt          -9.14947    4.15332  -2.203   0.0276 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8
```

PRACTICAL NO:- 7

Logistics Regression

AIM:- Practical of Logistics Regression

DESCRIPTION:- Logistic regression is a data analysis technique that uses mathematics to find the relationships between two data factors. It then uses this relationship to predict the value of one of those factors based on the other. The prediction usually has a finite number of outcomes, like yes or no.

PROGRAM:-

1. ```
   input <-mtcars[,c("am","cyl","hp","wt")]
   print(head(input))
   ```

2. ```
   input <-mtcars[,c("am","cyl","hp","wt")]
   am.data = glm(formula = am ~ cyl + hp + wt, data = input,
   family = binomial)
    print(summary(am.data))
   ```

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
Console   Terminal ×   Background Jobs ×
R 3.5.3 · C:/Users/cs/Desktop/Taslim DS/

> data1<-read.csv(file.choose(),sep=",",header = T)
> data1
   Time
1    85
2    95
3   105
4    85
5    90
6    97
7   104
8    95
9    88
10   90
11   94
12   95
```

PRACTICAL NO:- 8

Hypothesis Testing

AIM:- Practical of Hypothesis Testing.

DESCRIPTION:- Hypothesis testing is the process used to evaluate the strength of evidence from the sample and provides a framework for making determinations related to the population, ie, it provides a method for understanding how reliably one can extrapolate observed findings in a sample under study to the larger population.

PROGRAM:-

File:1 sample t test Hypo.csv

	A	B
1	Time	
2	85	
3	95	
4	105	
5	85	
6	90	
7	97	
8	104	
9	95	
10	88	
11	90	
12	94	
13	95	
14		

1. **"test for normal distribution"**
   data1<-read.csv(file.choose(),sep=",",header = T)

```
"test for normal distribution"
data1<-read.csv(file.choose(),sep=",",header = T)
```

2.

```
> shapiro.test(data1$Time)

        Shapiro-Wilk normality test

data:  data1$Time
W = 0.92653, p-value = 0.3448
```

3.

```
> apple<-read.csv(file.choose(),sep=",",header = T)
> apple
   Time
1    85
2    95
3   105
4    85
5    90
6    97
7   104
8    95
9    88
10   90
11   94
12   95
```

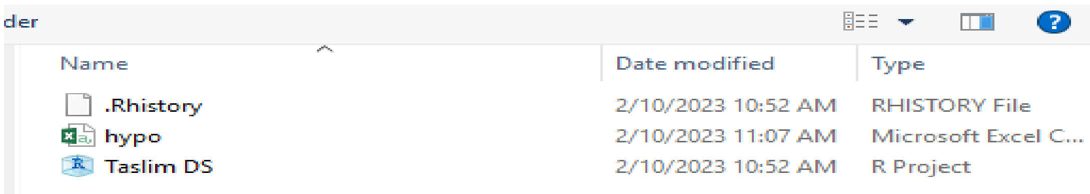
4.

```
> summary(apple)
      Time       
 Min.   : 85.00  
 1st Qu.: 89.50  
 Median : 94.50  
 Mean   : 93.58  
 3rd Qu.: 95.50  
 Max.   :105.00  
```

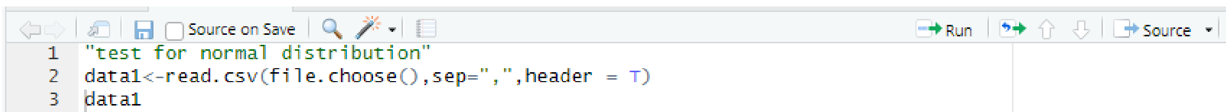
5.

```
> time1<-read.csv(file.choose(),sep=",",header = T)
> time1
   Time Time1
1    85    74
2    95    91
3    92    80
4   102    91
5    95    88
6    88    81
7    90    78
8    94    83
9    96    91
10   94    83
11  105   103
12   95    92
13   94    78
14   95    88
15   90    83
```

Select csv File



data1



2. **shapiro.test(data1$Time)**

3. **File: ONE SAMPLE t Test Hypo.csv**
   "one sample t test"
   apple<-read.csv(file.choose(),sep=",",header = T)
   apple

4. **summary(apple)**

5. **File: PAIRED t TEST.csv**

   "paired t test"

   time1<-read.csv(file.choose(),sep=",",header = T)

   time1

6.

```
> t.test(time1$Time,time1$Time1,alternative = "greater",paired = T)

        Paired t-test

data:  time1$Time and time1$Time1
t = 8.2143, df = 14, p-value = 5.027e-07
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 6.598881      Inf
sample estimates:
mean of the differences
                    8.4
```

7.

```
> cor<-read.csv(file.choose(),sep=",",header = T)
> cor
   Empcode aptitude job_prof
1     E101       86       88
2     E102       62       80
3     E103      110       96
4     E104      101       76
5     E105      100       80
6     E106       78       73
7     E107      120       58
8     E108      105      116
9     E109      112      104
10    E110      120       99
11    E111       87       64
12    E112      133      126
13    E113      140       94
14    E114       84       71
15    E115      106      111
16    E116      109      109
17    E117      104      100
18    E118      150      127
19    E119       98       99
20    E120      120       82
21    E121       74       67
22    E122       96      109
23    E123      104       78
24    E124       94      115
25    E125       91       83
```

8.

```
> summary(cor)
    Empcode      aptitude         job_prof    
 E101   : 1   Min.   : 62.0   Min.   : 58.0  
 E102   : 1   1st Qu.: 91.0   1st Qu.: 78.0  
 E103   : 1   Median :104.0   Median : 94.0  
 E104   : 1   Mean   :103.4   Mean   : 92.2  
 E105   : 1   3rd Qu.:112.0   3rd Qu.:109.0  
 E106   : 1   Max.   :150.0   Max.   :127.0  
 (Other):19                                  
```

6. **t.test(time1$time_before,time1$time_after,alternative = "greater",paired = T)**

7. **File: t test for correlation JOBPROF.csv**
   "t test for correlation " cor<-read.csv(file.choose(),sep=",",header = T)
   Cor

8. **summary(cor)**

9.

```
> cor.test(cor$aptitude,cor$job_prof,alternative = "two.sided",method="pearson")

        Pearson's product-moment correlation

data:  cor$aptitude and cor$job_prof
t = 2.8769, df = 23, p-value = 0.008517
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1497097 0.7558981
sample estimates:
      cor
0.5144107
```

10.

```
> time<-read.csv(file.choose(),sep=",",header = T)
> time
   Time_G1 Time_G2
1       85      83
2       95      85
3      105      96
4       85      94
5       90     102
6       97     100
7      104      94
8       95      95
9       88      88
10      90      92
11      94      95
12      95      94
13      NA      95
14      NA      90
```

11.

```
> summary(time)
    Time_G1          Time_G2
 Min.   : 85.00   Min.   : 83.00
 1st Qu.: 89.50   1st Qu.: 90.50
 Median : 94.50   Median : 94.00
 Mean   : 93.58   Mean   : 93.07
 3rd Qu.: 95.50   3rd Qu.: 95.00
 Max.   :105.00   Max.   :102.00
 NA's   :2
```

9. cor.test(cor$aptitude,cor$job_prof,alternative = "two.sided",method="pearson")

10. File: INDEPENDENT SAMPLES t TEST.csv
"independent t test "
time<-read.csv(file.choose(),sep=",",header = T)
time

11. summary(time)

12.

```
> var<-read.csv(file.choose(),sep=",",header = T)
> var
   Time_G1 Time_G2
1       85      83
2       95      85
3      105      96
4       85      94
5       90     102
6       97     100
7      104      94
8       95      95
9       88      88
10      90      92
11      94      95
12      95      94
13      NA      95
14      NA      90
```

13.

```
> summary(var)
    Time_G1          Time_G2      
 Min.   : 85.00   Min.   : 83.00  
 1st Qu.: 89.50   1st Qu.: 90.50  
 Median : 94.50   Median : 94.00  
 Mean   : 93.58   Mean   : 93.07  
 3rd Qu.: 95.50   3rd Qu.: 95.00  
 Max.   :105.00   Max.   :102.00  
 NA's   :2                        
```

14.

```
> var.test(var$Time_G1,var$Time_G2,alternative = "two.sided")

	F test to compare two variances

data:  var$Time_G1 and var$Time_G2
F = 1.5434, num df = 11, denom df = 13, p-value = 0.4524
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.4826988 5.2348866
sample estimates:
ratio of variances 
          1.543428 
```

12. "t test for variance "

```
var<-read.csv(file.choose(),sep=",",header = T)
var
```

13. summary(var)

14. var.test(var$Time_G1,var$Time_G2,alternative = "two.sided")

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
> data1<-read.csv(file.choose(),sep=",",header = T)
> data1
   satindex      dept
1        75   FINANCE
2        56   FINANCE
3        72   FINANCE
4        59   FINANCE
5        62   FINANCE
6        66   FINANCE
7        67   FINANCE
8        71   FINANCE
9        59   FINANCE
10       62   FINANCE
11       66   FINANCE
12       58   FINANCE
13       58 MARKETING
14       63 MARKETING
15       53 MARKETING
16       74 MARKETING
17       77 MARKETING
```

2.

```
> names(data1)
[1] "satindex" "dept"
```

3.

```
> summary(data1)
    satindex          dept   
 Min.   :53.00   FINANCE  :12
 1st Qu.:59.00   MARKETING: 5
 Median :63.00               
 Mean   :64.59               
 3rd Qu.:71.00               
 Max.   :77.00               
```

4.

```
> head(data1)
  satindex    dept
1       75 FINANCE
2       56 FINANCE
3       72 FINANCE
4       59 FINANCE
5       62 FINANCE
6       66 FINANCE
```

PRACTICAL NO:- 9

Analysis of Variance

AIM:- Practical of Analysis of Variance.

DESCRIPTION:- Analysis of variance, or ANOVA, is a statistical method that separates observed variance data into different components to use for additional tests. A one-way ANOVA is used for three or more groups of data, to gain information about the relationship between the dependent and independent variables.

PROGRAM:-

1. **Code:**
   ```
   "One way anova"
   data1<-read.csv(file.choose(),sep=",",header = T)
   data1
   ```

2. **names(data1)**

3. **summary(data1)**

4. **head(data1)**

5.

```
> anv<-aov(formula = satindex~dept,data = data1)
> anv
Call:
   aov(formula = satindex ~ dept, data = data1)

Terms:
                     dept Residuals
Sum of Squares     1.2010  828.9167
Deg. of Freedom         1        15

Residual standard error: 7.433782
Estimated effects may be unbalanced
```

6.

```
> summary(anv)
            Df Sum Sq Mean Sq F value Pr(>F)
dept         1    1.2    1.20   0.022  0.885
Residuals   15  828.9   55.26
```

7.

```
> data2<-read.csv(file.choose(),sep=",",header = T)
> data2
   satindex      dept exp
1        75   FINANCE lt5
2        56   FINANCE lt5
3        72   FINANCE lt5
4        59   FINANCE lt5
5        62   FINANCE lt5
6        66   FINANCE lt5
7        67   FINANCE gt5
8        71   FINANCE gt5
9        59   FINANCE gt5
10       62   FINANCE gt5
11       66   FINANCE gt5
12       58   FINANCE gt5
13       58 MARKETING lt5
14       63 MARKETING lt5
15       53 MARKETING lt5
16       74 MARKETING lt5
17       77 MARKETING lt5
```

8.

```
> names(data2)
[1] "satindex" "dept"     "exp"
```

5. **anv<-aov(formula = satindex~dept,data = data1)**
   anv

6. **summary(anv)**

7. **"Two way anova"**
   data2<-read.csv(file.choose(),sep=",",header = T)
   data2

8. **names(data2)**

9.

```
> summary(data2)
    satindex              dept      exp    
 Min.   :53.00   FINANCE  :12   gt5: 6  
 1st Qu.:59.00   MARKETING: 5   lt5:11  
 Median :63.00                          
 Mean   :64.59                          
 3rd Qu.:71.00                          
 Max.   :77.00                          
```

10.

```
> head(data2)
  satindex    dept exp
1       75 FINANCE lt5
2       56 FINANCE lt5
3       72 FINANCE lt5
4       59 FINANCE lt5
5       62 FINANCE lt5
6       66 FINANCE lt5
```

11.

```
> anv1<-aov(formula = satindex~dept+exp+dept,data = data2)
> summary(anv1)
            Df Sum Sq Mean Sq F value Pr(>F)
dept         1    1.2    1.20   0.020  0.889
exp          1    4.1    4.08   0.069  0.796
Residuals   14  824.8   58.92
```

9. names(data2)

10. head(data2)

11. anv1<-aov(formula = satindex~dept+exp+dept,data = data2)
summary(anv1)

CONCLUSION:- The above program has been executed successfully.

Outputs:-

1.

```
> fitness<-read.csv(file.choose(),sep=",",header=T)
> fitness
   Incomes GymVisits State Hours PayOrNot
1      100         4    TX   9.3      Yes
2       50         3    CA   4.8       No
3      100         4    TX   8.9      Yes
4      100         2    NY   6.5      Yes
5       50         2    MD   4.2       No
6       80         2    CA   6.2       No
7       75         3    WA   7.4       No
8       65         4    SD   6.0       No
9       90         3    ND   7.6      Yes
10      90         2    TX   6.1       No
11     100         1    TX   9.3      Yes
12      50         3    CA   4.8       No
13     100         4    TX   8.9       No
14     100         2    NY   6.5       No
15      50         2    MD   4.2       No
16      80         0    CA   6.2       No
17      75         3    WA   7.4       No
18      65         4    SD   6.0       No
19      90         3    ND   7.6      Yes
20      90         2    TX   6.1      Yes
21     100         4    TX   9.3      Yes
22      50         0    CA   4.8       No
23     100         4    TX   8.9       No
24     100         2    NY   6.5       No
25      50         2    TX   4.2       No
26      80         2    CA   6.2       No
27      75         3    TX   7.4       No
28      65         4    NY   6.0      Yes
29      90         2    MD   7.6      Yes
30      90         4    CA   6.1      Yes
```

2.

```
> names(fitness)
[1] "Incomes"   "GymVisits" "State"     "Hours"     "PayOrNot"
```

3.

```
> summary(fitness)
    Incomes         GymVisits       State       Hours         PayOrNot
 Min.   : 50.0   Min.   :0.000   CA:21   Min.   :4.200   No :54
 1st Qu.: 65.0   1st Qu.:2.000   MD: 8   1st Qu.:6.000   Yes:47
 Median : 90.0   Median :3.000   ND: 9   Median :6.500
 Mean   : 80.2   Mean   :2.931   NY:12   Mean   :6.704
 3rd Qu.:100.0   3rd Qu.:4.000   SD: 8   3rd Qu.:7.600
 Max.   :100.0   Max.   :7.000   TX:35   Max.   :9.300
                                 WA: 8
```

PRACTICAL NO:- 10

Decision Tree

AIM:- Practical of Decision Tree.

DESCRIPTION:- A decision tree is one of the supervised machine learning algorithms. This algorithm can be used for regression and classification problems — yet, is mostly used for classification problems. A decision tree follows a set of if-else conditions to visualize the data and classify it according to the conditions.

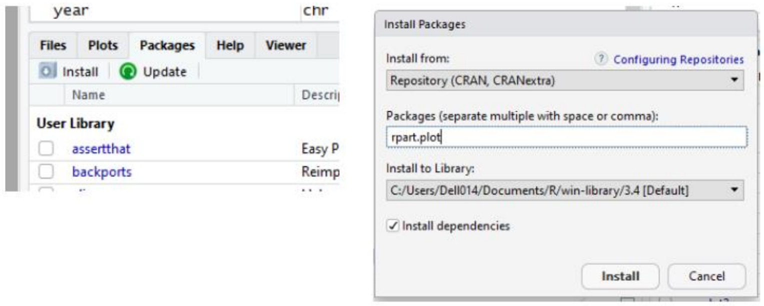
PROGRAM:-

1. **fitness<-read.csv(file.choose(),sep=",",header=T)**
   fitness

2. **names(fitness)**

3. **summary(fitness)**

4.




```
installing the source package 'rpart.plot'

trying URL 'https://cran.rstudio.com/src/contrib/rpart.plot_3.1.1.tar.gz'
Content type 'application/x-gzip' length 672084 bytes (656 KB)
downloaded 656 KB

* installing *source* package 'rpart.plot' ...
** package 'rpart.plot' successfully unpacked and MD5 sums checked
** R
** data
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
  converting help for package 'rpart.plot'
    finding HTML links ... done
    prp                                     html
    ptitanic                                html
    rpart.plot                              html
    rpart.predict                           html
    rpart.rules                             html
    show.prp.palettes                       html
** building package indices
** testing if installed package can be loaded
*** arch - i386
*** arch - x64
* DONE (rpart.plot)
In R CMD INSTALL
```

5.

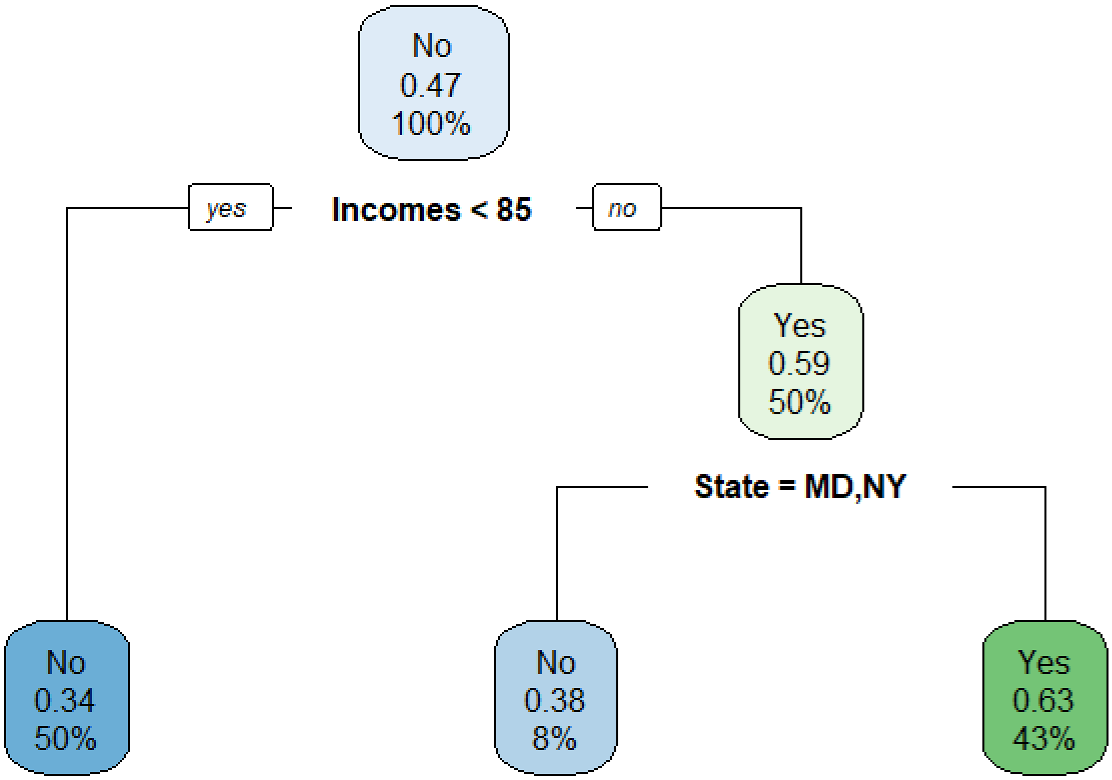
```
> library(rpart)
> treeAnalysis<-rpart(PayOrNot~Incomes+GymVisits+State,data=fitness)
> treeAnalysis
n= 101

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 101 47 No (0.5346535 0.4653465)
  2) Incomes< 85 50 17 No (0.6600000 0.3400000) *
  3) Incomes>=85 51 21 Yes (0.4117647 0.5882353)
    6) State=MD,NY 8  3 No (0.6250000 0.3750000) *
    7) State=CA,ND,SD,TX,WA 43 16 Yes (0.3720930 0.6279070) *
```

4. **Install library(rpart.plot)**

5. **library(rpart)**
   treeAnalysis<-rpart(PayOrNot~Incomes+GymVisits+State,data=fitness)
   treeAnalysis

6.

```
> library("rpart.plot")
> rpart.plot(treeAnalysis)
```



6. **library("rpart.plot")**
   rpart.plot(treeAnalysis

CONCLUSION:- The above program has been executed successfully.