THAKUR COLLEGE OF SCIENCE & COMMERCE

NAAC Accredited with Grade "A" (3rd Cycle)

THAKUR TRUSTS

ISO 9001 : 2015 Certified

Degree College

Computer Journal

CERTIFICATE

SEMESTER **6** UID No. 2020878

Class TYBSC(CS) Roll No. 380 Year 2022-2023

This is to certify that the work entered in this journal is the work of Mst. / Sumit Singh

who has worked for the year 2022-2023 in the Computer Laboratory.

Teacher In-Charge

Head of Department

Date :

Examiner

Index

Prac No	Aim of practical	Page No
1	Practical of Data collection, Data curation and management for Large-scale Data system (such as MongoDB) CRUD operations using MongoDB	**3**
2	Demonstration of Simple and Multiple Linear Regression (Using Python or R)	**15**
3	Demonstration of Logistics Regression (Using Python / R)	**22**
4	Demonstration of Hypothesis testing (Using Python or R)	**29**
5	Demonstration of Analysis of Variance (Using Python or R)	**32**
6	Demonstration of Decision Tree (Using Python or R)	**36**
7	Demonstration of Principal Component Analysis (Using Python or R)	**40**
8	Demonstration of Clustering (Using Python or R)	**45**
9	Demonstration of Time-series forecasting (Using Python or R)	**49**
10	Use of R Markdown and RStudio Cloud (Store mini project in RStudio Cloud)	**55**

Practical 1

Aim: Data collection, Data curation and management for Large-scale Data system (such as MongoDB) CRUD operations using MongoDB

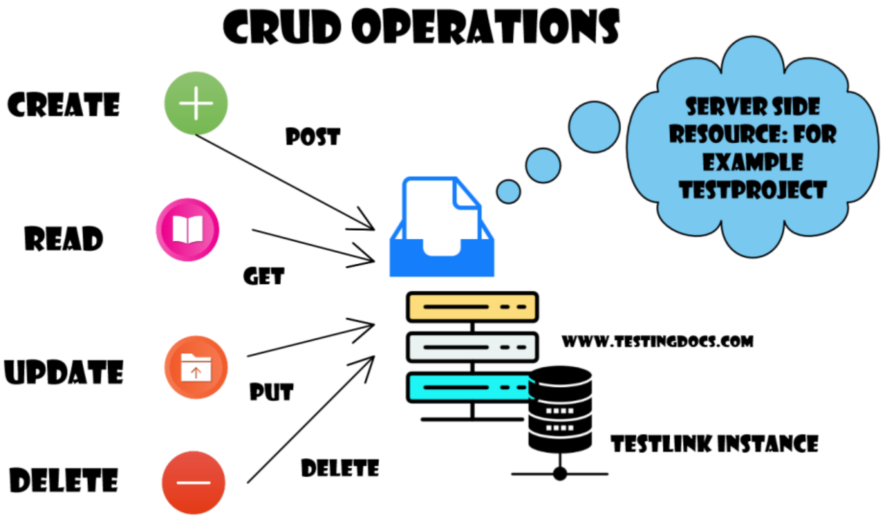
Theory:

MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time

- The document model maps to the objects in your application code, making data easy to work with
- Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
- MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
- MongoDB is free to use. Versions released prior to October 16, 2018 are published under the AGPL. All versions released after October 16, 2018, including patch fixes for prior versions, are published under the Server-Side Public License (SSPL) v1.

Within computer programming, the acronym CRUD stands for create, read, update, and delete. These are the four basic functions of persistent storage. Also, each letter in the acronym can refer to all functions executed in relational database applications and mapped to a standard HTTP method, SQL statement, or DDS operation.

It can also describe user-interface conventions that allow viewing, searching, and modifying information through computer-based forms and reports. In essence, entities are read, created, updated, and deleted. Those same entities can be modified by taking the data from a service and changing the setting properties before sending the data back to the service for an update. Plus, CRUD is data-oriented and the standardized use of HTTP action verbs.



Code:

Starting server with mongo or mongodb

C: \ >mongo >db Test

- Create Database In MongoDB Once you are in the MongoDB shell, create the database in MongoDB by typing this command:

```
use database_name
```

For example: create a database “tycs”:

> use tycs switched to db tycs

```
MongoDB Enterprise > use tycs
switched to db tycs
MongoDB Enterprise > show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
tycs      0.000GB
MongoDB Enterprise >
```

Create a collection user and insert a document in it.

> db.user.insert((name: "Asif", age: 20})

O/P: WriteResult(("nInserted" : 1 })

>showdbs admin

0.000GB config

0.000GB local 0.000GB

tycs 0.000GB

• MongoDB Drop Database

```
>db.dropDatabase()
```

O/P:

("dropped" : "Testdb", "ok" : 1 }

MongoDB Enterprise > show dbs

admin 0.000GB config 0.000GB

local 0.000GB tycs 0.000GB O/P:

```
MongoDB Enterprise > db.dropDatabase()
{ "dropped" : "tycs", "ok" : 1 }
MongoDB Enterprise > show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
MongoDB Enterprise >
```

• Create Co1Iect1ozz 1zz MozzgoDB

Method 1: Creatlng the Collection 1n MongoDB on the

fly MongoDB Enterprise > use tycs switched to db tycs

MongoDB Enterprise > db.tycs.insert((name:"Asif khan",age:21,website:"www.goog1e.com"})

WriteResult(("nInserted" : 1 })

Syntax: db.co1leetion_name.find{J

MongoDB Enterprise > db.tycs.find()

o/p: ("_id" : ObjectId("5e410808e3755b1e06a63d1d"), "name" : "Asif khan", "age" : 21, "website" : "www.google.com" }

```
show collections
```

MongoDB Enterprise > show

collections

O/P: tycs user

• Drop collection Ie MongoDB

SYNTAX

```
db.co11ection_name.drop()
```

MongoDB Enterprise > use students

switched to db students MongoDB

Enterprise > show collections students

teachers tycs user

MongoDB Enterprise > db.user.drop{J true

MongoDB Enterprise > show collections students

Teacher

tycs

MongoDB Insert Document

Syntax to insert a document into the collection:

```
db.collection_name.insert()
```

> db.tycs.insert{

zzazzze: "ASIE'", age: 20,

email: "as1f@gmai1.com",

... course: [(zzame: "MozzgoDB", durataozz: 7 },

(zzame: "Java", duration: 30 }]

O P: WriteResult(("nInserted" : 1 })

Verification:

Syntax:

```
db.co11eetion_name.find{}
```

> db.tycs.find{J

(" id" : Objectld("5c2d37734fa204bd77e7fc1c"), "name" : "ASIF", "age" : 20,

"email" : "asif@gmail.com", "course" : [{ "name" :

"MongoDB", "duration" : 7 },

{ "name" : "Java", "duration" : 30 }] }

MongoDB Example: Insert Multiple Documents in collection

MongoDB Enterprise > var beginners=

... "studentID": 1001,

... "studentName":"Asif",

... "age":20

• MongoDB Query Document using fiiid{J method Querying all the documents in JSON format

MongoDB Enterprise > db.students.find().pretty()

"_id" ObjectId("5e4lof3fe3755bleo6a63d1e"),

"studentID" : 1001,

"studentName" : "Asif",

"age" : 20

• Quezy Document based on tfze criteria

> db.students.find({StudentName "Asif'}).pretty() "

_id" ObjectId("5c28lc90c23e08d l5l5fd9cc"),

"Studentld" : 1001,

"StudentName" : "ASlf",

"age" : 20

• Updating Document using update() method

Syntax:

```
db.co11ection_name.update(criteria, update_data)
```

> use tycs switched to db

tycs > sbow co11ectlozzs

beginnersbook students

tycs

> db.createCol1eetion{"got"} (

"ok" : 1 }

> var abc = [

"_id" ObjectId("59bd2e73ce524b733fl4dd65"), "name" "Asifi",

"age" 20 db.co11ection_name.update(criteria, update_data)

> db.got.find(J.pretty(J " id" Objectld("59bd2e73ce524b733fl4dd65"),

"name" : "steve",

"age" : 20

To update multiple documents with the update{} method

```
db.got.update(("name":"Jon Snow"},
 ($ set :("name":"Kit Harington"}},(multi:true})
```

Updating Document using save() method Syntax:

```
db.collection_name.save( (_id:ObjectId(), new_document} )
```

```
db.got.find().pretty()
```

> db.got.find(("name”: "Asifi'}).pretty()

"did" Objectld("59bd2e73ce524b733fl4dd65"),

"name" : "Asifi’,

"age" : 20

> db.got.find().pretty()

"_id" ObjectId("59bd2e73ce524b733fl4dd65"),

"name" : "Steve",

"age" : 20

• MongoDB Delete Document from a Collection

Syntax of remove(J method

```
db.collection_name.remove(de1ete_criteria)
```

Delete Document using remove{J method

```
> db.students.find().pretty()

    " id"    ObjectId("59bcecc7668dcce02aaa6fed"),
    "StudentId" : 1001,
    "StudentName" : "Steve",
    "age" : 30
```

```
db.students.remove(("StudentId": 3333})
```

Output:

WriteResult(("nRemoved" : 1 }) ***To verify whether the document is actually deleted. Type the following command:***

```
db.students.find().pretty()
```

It will list all the documents of students collection.

> use tycs switched to db

tycs

> db.students.find().pretty()

" id" Objectld("5c28lc90c23e08d15 l5fd9cc"),

"Studentld" : 1001, "StudentName" : "Asif",

"age" : 20 "_id" ObjectId("5c2d38934fa204bd77e7fc1d"),

"Studentld" : 1001, "StudentName" : "Steve",

"age" : 30

Removes all Documents

- ***MongoDB Projection***

Syntax:

```
db.co11ection_name.find(§,(fie1d_key:1 or 0})
```

> db.students.find().pretty()

"_id" Objectld("5c28lc90c23e08d15 l5fd9cc"),

"Studentld" : 1001,

"StudentName" : "Steve",

"age" : 20

> db.students.find(§, ("_id": 0, "StudentId" : 1})
("StudentId" : 1001 } (
"StudentId" : 1002 }

> db.students.find(§, ("_id": 0, "StudentName" : 0, "age" : 0})
("Studentld" : 1001 }
{ "Studentld" : 1002 }

• MongoDB — limit(J and ship{ J method The limit(J method in MongoDB

Syntax:

```
db.collection_name.find().limit(number_of_documents)
db.studentdata.find((student_id  ($gt:2002}}).pretty()
db.studentdata.find((student_id  ($gt:2002}}).limit(1).pretty()
```

]

```
db.studentdata.find((studentpid  {$gt:2002}}).limit(1).skip(1).pretty()
```

MongoDB Sk1p() Method

• tongoDB aort() method Sorting Documenta ualng aort() method Syntax o£ sort() method:

```
db.col1ecttion_name.find().sort({field_key:1 or -1})
```

1 is for ascending order and -1 is for descending order.

The default value 1s 1

. For example: collection studentdata contains following documents:

```
> db.studentdata.find().pretty()

    "_id"    ObjectId("59bf63380be ld7770c3982afi'),
    "student_name" : "Steve",
    "student id" : 1001, "student age"
    :1002
```

Let's display the[S t, ldent_id of all the documents in descending order

```
> db.studentdata.find(§, {"student_id": 1, _id:0}).sort({"student_id": -1})
( "student_id" : 1001 }
( "student_id" : 1002 }
```

To display the studentqid field of all the students in ascending order:

```
> db.studentdata.find(§, {"student_id": 1, did:0}).sort({"student_id": 1})
( "student_id" : 1001 }
( "student id" : 1002 }
```

```
> db.studentdata.find({}, {"student_id": 0, _id:0}).sort({"st
udent_id": 1})
{ "student_name" : "Steve", "student_age" : 22 }
{ "student_name" : "Carol", "student_age" : 22 }
{ "student_name" : "Tim", "student_age" : 23 }
>
```

• MongoDB Indexing with Example How

to create index in MongoDB

db.collection_name.createIndex((field_name: 1 or -1})

The value 1 is for ascending order and -1 is for descending order.

Let's create the index on student_name field in ascending order:

db.studentdata.createIndex((student_name: 1})

```
"createdCollectionAutomatically" :false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
..p..
```

- *MongoDB — Finding the indexes in a collection*

```
db.collection_name.getIndexes()
> db.studentdata.getIndexes()
```

```
"v"   2,                                          )
"key"   (
       "_id"   1

"name" : " id ",
"ns" : "test studentdata"
```

Practical 2

Aim: Demonstration of Simple and Multiple Linear Regression.

Theory:

Simple linear regression is used to estimate the relationship between two quantitative variables. You can use simple linear regression when you want to know:

How strong the relationship is between two variables (e.g., the relationship between rainfall and soil erosion).

The value of the dependent variable at a certain value of the independent variable (e.g., the amount of soil erosion at a certain level of rainfall).

Regression models describe the relationship between variables by fitting a line to the observed data. Linear regression models use a straight line, while logistic and nonlinear regression models use a curved line. Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.

Regression models are used to describe relationships between variables by fitting a line to the observed data. Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable.

Code1:

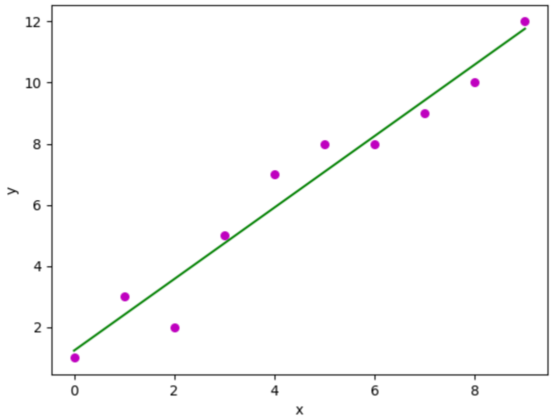
```
import numpy as np

import matplotlib.pyplot as plt


def estimate_coef(x, y):

    # number of observations/points

    n = np.size(x)


    # mean of x and y vector

    m_x = np.mean(x)

    m_y = np.mean(y)


    # calculating cross-deviation and deviation about x

    SS_xy = np.sum(y*x) - n*m_y*m_x

    SS_xx = np.sum(x*x) - n*m_x*m_x


    # calculating regression coefficients

    b_1 = SS_xy / SS_xx

    b_0 = m_y - b_1*m_x


    return (b_0, b_1)


def plot_regression_line(x, y, b):

    # plotting the actual points as scatter plot

    plt.scatter(x, y, color = "m",

                marker = "o", s = 30)
```

```
    # predicted response vector
    y_pred = b[0] + b[1]*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')

    # function to show plot
    plt.show()

def main():
    # observations / data
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} \
        \nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)
```

```
if __name__ == "__main__":
    main()
```

Output1:

```
Estimated coefficients:
b_0 = 1.2363636363636363
b_1 = 1.1696969696969697
```



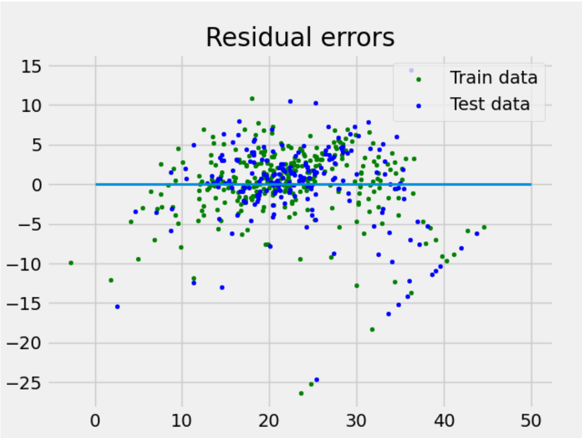
Code2:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics

# load the boston dataset
boston = datasets.load_boston(return_X_y=False)

# defining feature matrix(X) and response vector(y)
X = boston.data
y = boston.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=1)

# create linear regression object
reg = linear_model.LinearRegression()

# train the model using the training sets
reg.fit(X_train, y_train)

# regression coefficients
print('Coefficients: ', reg.coef_)
```

```
# variance score: 1 means perfect prediction
print('Variance score: {}'.format(reg.score(X_test, y_test)))

# plot for residual error

## setting plot style
plt.style.use('fivethirtyeight')

## plotting residual errors in training data
plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train,
            color = "green", s = 10, label = 'Train data')

## plotting residual errors in test data
plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test,
            color = "blue", s = 10, label = 'Test data')

## plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2)
## plotting legend
plt.legend(loc = 'upper right')
## plot title
plt.title("Residual errors")
## method call for showing the plot
plt.show()
```

Output2:

```
Coefficients:  [-8.95714048e-02  6.73132853e-02  5.04649248e-02  2.18579583e+00
 -1.72053975e+01  3.63606995e+00  2.05579939e-03 -1.36602886e+00
  2.89576718e-01 -1.22700072e-02 -8.34881849e-01  9.40360790e-03
 -5.04008320e-01]
Variance score: 0.7209056672661767
```



Conclusion:

Multiple linear regression is a more specific calculation than simple linear regression. For straightforward relationships, simple linear regression may easily capture the relationship between the two variables. For more complex relationships requiring more consideration, multiple linear regression is often better.

Practical 3

Aim: Demonstration of Logistics Regression.

Theory:
Logistics regression is also known as generalized linear model. As it is used as a classification technique to predict a qualitative response, Value of y ranges from 0 to 1 and can be represented by following equation:

$$\text{Odds} = \frac{P}{1 - P}$$

p is probability of characteristic of interest. The odds ratio is defined as the probability of success in comparison to the probability of failure. It is a key representation of logistic regression coefficients and can take values between 0 and infinity. Odds ratio of 1 is when the probability of success is equal to the probability of failure. Odds ratio of 2 is when the probability of success is twice the probability of failure. Odds ratio of 0.5 is when the probability of failure is twice the probability of success.

$$\log(\text{Odds}) = \log\left(\frac{P}{1 - P}\right)$$

Since we are working with a binomial distribution(dependent variable), we need to choose a link function that is best suited for this distribution.

$$\text{logit}(P) = \log\left(\frac{P}{1 - P}\right) = b_0 + b_1{*}1 + b_2{*}2 + b_3{*}3 + \ldots + b_k{*}k$$

It is **logit function**. In the equation above, the parenthesis is chosen to maximize the likelihood of observing the sample values rather than minimizing the sum of squared errors(like ordinary regression). The logit is also known as a log of odds. The logit function must be linearly related to the independent variables. This is from equation A, where the left-hand side is a linear combination of x. This is similar to the OLS assumption that y be linearly related to x.
Variables b0, b1, b2 ... etc are unknown and must be estimated on available training data. In a logistic regression model, multiplying b1 by one unit changes the logit by b0. The P changes due to a one-unit change will depend upon the value multiplied. If b1 is positive then P will increase and if b1 is negative then P will decrease.

Code:

X<-read.csv("C:/Users/Admin/Documents/SampleStudentData.csv")

> X

```
R Console

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> x=read.csv("d:/weather3.csv")
> x
    outlook temperature humidity windy play
1  overcast         hot     high FALSE  yes
2  overcast        cool   normal  TRUE  yes
3  overcast        mild     high  TRUE  yes
4  overcast         hot   normal FALSE  yes
5     rainy        mild     high FALSE  yes
6     rainy        cool   normal FALSE  yes
7     rainy        cool   normal  TRUE   no
8     rainy        mild   normal FALSE  yes
9     rainy        mild     high  TRUE   no
10    sunny         hot     high FALSE   no
11    sunny         hot     high  TRUE   no
12    sunny        mild     high FALSE   no
13    sunny        cool   normal FALSE  yes
14    sunny        mild   normal  TRUE  yes
> 
```

PRINTING THE DATASET

>x$humidity=ifelse(test=x$humidity=="high",yes=1,no=0)

>x

```
> x$humidity=ifelse(test=x$humidity=="high",yes=1,no=0)
> x
    outlook temperature humidity windy play
1  overcast         hot        1 FALSE  yes
2  overcast        cool        0  TRUE  yes
3  overcast        mild        1  TRUE  yes
4  overcast         hot        0 FALSE  yes
5     rainy        mild        1 FALSE  yes
6     rainy        cool        0 FALSE  yes
7     rainy        cool        0  TRUE   no
8     rainy        mild        0 FALSE  yes
9     rainy        mild        1  TRUE   no
10    sunny         hot        1 FALSE   no
11    sunny         hot        1  TRUE   no
12    sunny        mild        1 FALSE   no
13    sunny        cool        0 FALSE  yes
14    sunny        mild        0  TRUE  yes
```

>x$play=ifelse(test=x$play=="yes",yes=1,no=0)

>x

```
> x$play=ifelse(test=x$play=="yes",yes=1,no=0)
> x
    outlook temperature humidity windy play
1  overcast         hot        1 FALSE    1
2  overcast        cool        0  TRUE    1
3  overcast        mild        1  TRUE    1
4  overcast         hot        0 FALSE    1
5     rainy        mild        1 FALSE    1
6     rainy        cool        0 FALSE    1
7     rainy        cool        0  TRUE    0
8     rainy        mild        0 FALSE    1
9     rainy        mild        1  TRUE    0
10    sunny         hot        1 FALSE    0
11    sunny         hot        1  TRUE    0
12    sunny        mild        1 FALSE    0
13    sunny        cool        0 FALSE    1
14    sunny        mild        0  TRUE    1
```

>x$windy=ifelse(test=x$windy=="FALSE",yes=0,no=1)
>x

```
> x$windy=ifelse(test=x$windy=="FALSE",yes=0,no=1)
> x
    outlook temperature humidity windy play
1  overcast         hot        1     0    1
2  overcast        cool        0     1    1
3  overcast        mild        1     1    1
4  overcast         hot        0     0    1
5     rainy        mild        1     0    1
6     rainy        cool        0     0    1
7     rainy        cool        0     1    0
8     rainy        mild        0     0    1
9     rainy        mild        1     1    0
10    sunny         hot        1     0    0
11    sunny         hot        1     1    0
12    sunny        mild        1     0    0
13    sunny        cool        0     0    1
14    sunny        mild        0     1    1
> |
```

PARTIONING DATASET
> s=sample(nrow(x),.7*nrow(x))
>x_tr=x[s,]
>x_test=x[-s,]
>nrow(x)
>nrow(x_tr)
>nrow(x_test)

```
> s=sample(nrow(x),.7*nrow(x))
> x_tr=x[s,]
> x_test=x[-s,]
> nrow(x)
[1] 14
> nrow(x_tr)
[1] 9
> nrow(x_test)
[1] 5
> |
```

Data Modeling
>lmod=glm(play~windy,data=x_tr,family=binomial,control=list(maxit=100))
>lmod

```
> lmod=glm(play~windy,data=x_tr,family=binomial,control=list(maxit=100))
> lmod

Call:  glm(formula = play ~ windy, family = binomial, data = x_tr, control = list(maxit = 100))

Coefficients:
(Intercept)        windy
      20.57       -19.87

Degrees of Freedom: 8 Total (i.e. Null);  7 Residual
Null Deviance:      6.279
Residual Deviance: 3.819        AIC: 7.819
> |

> summary(lmod)

Call:
glm(formula = play ~ windy, family = binomial, data = x_tr, control = list(maxit = 100))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.48230   0.00005   0.00005   0.00005   0.90052

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    20.57    7238.39   0.003    0.998
windy         -19.87    7238.39  -0.003    0.998

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.2790  on 8  degrees of freedom
Residual deviance: 3.8191  on 7  degrees of freedom
AIC: 7.8191

Number of Fisher Scoring iterations: 19

> |
```

>lmod=glm(play~humidity,data=x_tr,family=binomial,control=list(maxit=100))
>summary(lmod)

```
> lmod=glm(play~humidity,data=x_tr,family=binomial,control=list(maxit=100))
> summary(lmod)

Call:
glm(formula = play ~ humidity, family = binomial, data = x_tr,
    control = list(maxit = 100))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.97277   0.00008   0.55525   0.55525   0.55525

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.792      1.080   1.659   0.0971 .
humidity      17.774   7604.236   0.002   0.9981
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.2790  on 8  degrees of freedom
Residual deviance: 5.7416  on 7  degrees of freedom
AIC: 9.7416

Number of Fisher Scoring iterations: 18

> |
```

>lmod=glm(play~temperature,data=x_tr,family=binomial,control=list(maxit=100))
>summary(lmod)

```
> lmod=glm(play~temperature,data=x_tr,family=binomial,control=list(maxit=100))
> summary(lmod)

Call:
glm(formula = play ~ temperature, family = binomial, data = x_tr,
    control = list(maxit = 100))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.66511   0.00005   0.00005   0.75853   0.75853

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)         1.099      1.155   0.951    0.341
temperaturehot     19.467  12537.265   0.002    0.999
temperaturemild    19.467  10236.634   0.002    0.998

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.2790  on 8  degrees of freedom
Residual deviance: 4.4987  on 6  degrees of freedom
AIC: 10.499

Number of Fisher Scoring iterations: 19

> |
```

Prediction:

> p=predict(lmod,x_test,type="response")

>p

```
> p=predict(lmod,x_test,type="response")
> p
           3            9           10           11           12
1.000000e+00 5.800756e-11 1.000000e+00 1.000000e+00 1.000000e+00
> |
```

(2) Second data set:

#IMPORT THE DATA

>x2=read.csv("D:/grade_logit.csv")

>x2

```
> x2=read.csv("D:/grade_logit.csv")
> x2
   Exam1 Exam2 Exam3 Exam4 Final_score Grade
1     60    10    16   7.0       40.79     1
2     90     0     0   0.0       69.23     1
3    130    20    24   1.0       76.75     1
4    130    10    24   8.5       75.66     1
5     90     5    22   9.5       55.48     1
6    100    30    20   3.0       67.11     1
7    105    20    22   8.0       67.98     1
8    120    40    18  16.0       85.09     1
9    120    20    30  18.0       82.46     1
10   130    45    22  10.5       91.01     1
11    90    40    20   7.0       68.86     1
12   130    30    28  10.5       87.06     1
13   100    30    22   6.5       69.52     1
14     0    30    18   0.0       60.00     1
15     0    30    18   0.0       60.00     1
16    80     0    24   3.0       60.11     1
17   105    40    22   6.5       76.10     1
18    10     0     0   8.0       12.16     0
19   130    35    24   0.0       90.00     1
20     0    15    20   7.0       42.86     1
21    40    10    14   6.0       30.70     0
22    90    15    28   8.5       62.06     1
23   110     0    24   9.5       80.62     1
24    65     5    24   1.0       41.67     1
25    55    15    18   0.0       41.90     1
26   100    50    30  11.5       83.99     1
27    95    40    24   8.0       73.25     1
28     0    10    24   0.0       42.50     1
29     0     0    18   0.0       60.00     1
30    65    20    20   0.0       50.00     1
31   110    25    18   6.0       69.74     1
32   130    45    24   8.0       90.79     1
33   120    40    30   9.0       87.28     1
34    70    20    24   1.0       50.44     1
35   130    45    10  16.5       88.38     1
```

>

lmod2=glm(Grade~Exam1,data=x2_train,family=binomial,control=list(maxit=100)
)
>summary(lmod2)

```
> lmod2=glm(Grade~Exam1,data=x2_train,family=binomial,control=list(maxit=100))
> summary(lmod2)

Call:
glm(formula = Grade ~ Exam1, family = binomial, data = x2_train,
    control = list(maxit = 100))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2051   0.1834   0.2442   0.4444   0.9351

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.600860   0.396710   1.515  0.12987
Exam1       0.028971   0.009424   3.074  0.00211 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 68.589  on 82  degrees of freedom
Residual deviance: 54.049  on 81  degrees of freedom
AIC: 58.049

Number of Fisher Scoring iterations: 6
```

Prediction data 1's and 0's form
>prediction=ifelse(p>.5,1,0)
>prediction

```
> prediction=ifelse(p>.5,1,0)
> prediction
 4 10 13 14 23 37 45 50 51 55 64 66 67 76 81 84 89 91 93 96 97
 1  1  1  1  1  1  1  1  0  1  1  1  0  1  1  1  0  1  1  1  1
>
```

PREDICTION MATRIX
>table(x2_test$Grade,prediction)

```
> table(x2_test$Grade,prediction)
   prediction
     0  1
  0  2  1
  1  1 17
```

> x2_test

```
> x2_test
   Exam1 Exam2 Exam3 Exam4 Final_score Grade
4    130    10    24   8.5       75.66     1
10   130    45    22  10.5       91.01     1
13   100    30    22   6.5       69.52     1
14     0    30    18   0.0       60.00     1
23   110     0    24   9.5       80.62     1
37     0    25    24   0.0       61.25     1
45    95    30    30  12.0       73.25     1
50   130    40    28  16.5       94.08     1
51     0     0     0  15.5       86.11     1
55   110    25    20   3.0       69.30     1
64   125    30    30  11.5       86.18     1
66    75    15    16   0.0       50.48     1
67     0     0     0   5.0       27.78     0
76   100    35    24   0.0       75.71     1
81    50    20    20   1.0       39.91     0
84   100    35    24  10.5       74.34     1
89     0     0     0   2.0       11.11     0
91   110    25    24   4.0       71.49     1
93    85    30    20   2.5       60.31     1
96   100    35    20   0.0       73.81     1
97     0     0    26   0.0       86.67     1
>
```

#actuals predicted
>ac_pr<- data.frame(cbind(actuals=x2_test$Grade, predicteds=prediction))
>ac_pr

```
> ac_pr <- data.frame(cbind(actuals=x2_test$Grade, predicteds=prediction))
> ac_pr
   actuals predicteds
4        1          1
10       1          1
13       1          1
14       1          1
23       1          1
37       1          1
45       1          1
50       1          1
51       1          0
55       1          1
64       1          1
66       1          1
67       0          0
76       1          1
81       0          1
84       1          1
89       0          0
91       1          1
93       1          1
96       1          1
97       1          1
>
```

>vif(lmod2) // variable influence factor

```
> vif(lmod2)
   Exam1    Exam2    Exam3 
1.023350 1.117704 1.122152 
>
```

Conclusion: logistic regression is used for classification problems when the output or dependent variable is dichotomous or categorical.

Practical 4

Aim: Demonstration of Hypothesis testing.

Theory:

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 from which the sample was drawn. The investigator formulates a specific hypothesis, evaluates data from the sample, and uses these data to decide whether they support the specific hypothesis.

The first step in testing hypotheses is the transformation of the research question into a null hypothesis, H0, and an alternative hypothesis, HA. 6 The null and alternative hypotheses are concise statements, usually in mathematical form, of 2 possible versions of "truth" about the relationship between the predictor of interest and the outcome in the population. These 2 possible versions of truth must be exhaustive (ie, cover all possible truths) and mutually exclusive (ie, not overlapping). The null hypothesis is conventionally used to describe a lack of association between the predictor and the outcome; the alternative hypothesis describes the existence of an association and is typically what the investigator would like to show. The goal of statistical testing is to decide whether there is sufficient evidence from the sample under study to conclude that the alternative hypothesis should be believed.

Hypothesis testing has been likened to a criminal trial, in which a jury must use evidence to decide which of 2 possible truths, innocence (H0) or guilt (HA), is to be believed. Just as a jury is instructed to assume that the defendant is innocent unless proven otherwise, the investigator should assume there is no association unless there is strong evidence to the contrary. A jury's verdict must be either guilty or not guilty, in which case a not-guilty verdict does not equal innocence. Rather, it indicates that the burden of proof has not been met. Similarly, an investigator can only reject H0 or fail to reject it; failure to reject does not prove that the null H0 is true.

Code:

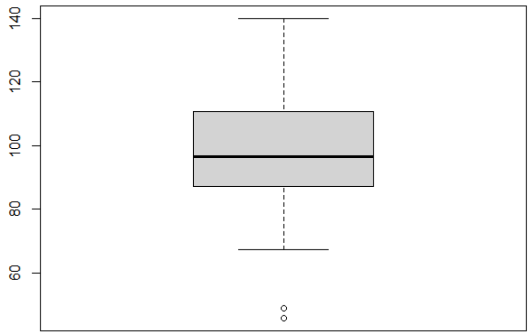
```
data=read.csv("D:/College/Sumit/DS/prac4.csv")

data

boxplot(data)

m1=mean(data$C1)

m1

sd1=sd(data$C1)

sd1

plot(data$C1)

t.test(data$C1,alternative="greater",mu=100)
```

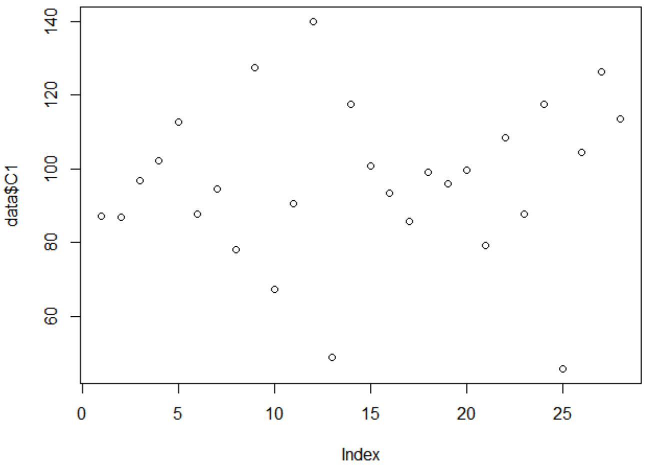
Output:

```
> data=read.csv("D:/College/Sumit/DS/prac4.csv")
> data
      C1
1   87.1
2   86.9
3   96.8
4  102.2
5  112.6
6   87.7
7   94.5
8   78.0
9  127.5
10  67.3
11  90.6
12 140.0
13  48.9
14 117.5
15 100.8
16  93.2
17  85.7
18  98.9
19  96.0
20  99.6
21  79.1
22 108.5
23  87.6
24 117.5
25  45.6
26 104.4
27 126.2
28 113.4
> boxplot(data)
> m1=mean(data$C1)
> m1
[1] 96.21786
> sd1=sd(data$C1)
> sd1
[1] 21.33573
```

```
> plot(data$C1)
> t.test(data$C1,alternative="greater",mu=100)

        One Sample t-test

data:  data$C1
t = -0.93801, df = 27, p-value = 0.8217
alternative hypothesis: true mean is greater than 100
95 percent confidence interval:
 89.35007      Inf
sample estimates:
mean of x
 96.21786
```

Graphs:





Conclusion:

Thus we have implementedHypothesis testing of a Single Population means successfully

Practical 5

Aim: Demonstration of Analysis of Variance

Theory:
Analysis of variance (ANOVA) is an analysis tool used in statistics that splits an observed aggregate variability found inside a data set into two parts: systematic factors and random factors. The systematic factors have a statistical influence on the given data set, while the random factors do not. Analysts use the ANOVA test to determine the influence that independent variables have on the dependent variable in a regression study.
There are two types of ANOVA tests: one-way ANOVA and two-way ANOVA. In a one-way ANOVA test, there is one independent variable and one dependent variable. A one-way ANOVA test would be used, for example, to observe which diet caused the most weight loss in individual participants. In this study, the different diets (vegan, vegetarian, keto, etc.) would be the independent variables. The dependent variable is the amount of weight lost. In a two-way ANOVA test, there is still one dependent variable, but there are two independent variables. Referring to the same example of diet and weight loss, a twoway ANOVA would be used to measure the combination of different diets and exercises on weight loss. In this study, the dependent variable is still weight loss, but there are now two different independent variables (diet and exercise).

The formula for ANOVA is F=MST/MSE , where:

- F is ANOVA.
- MST is the mean of the sum of the squares due to treatment.
- MSE is the mean of the sum of the squares due to error.

Code:

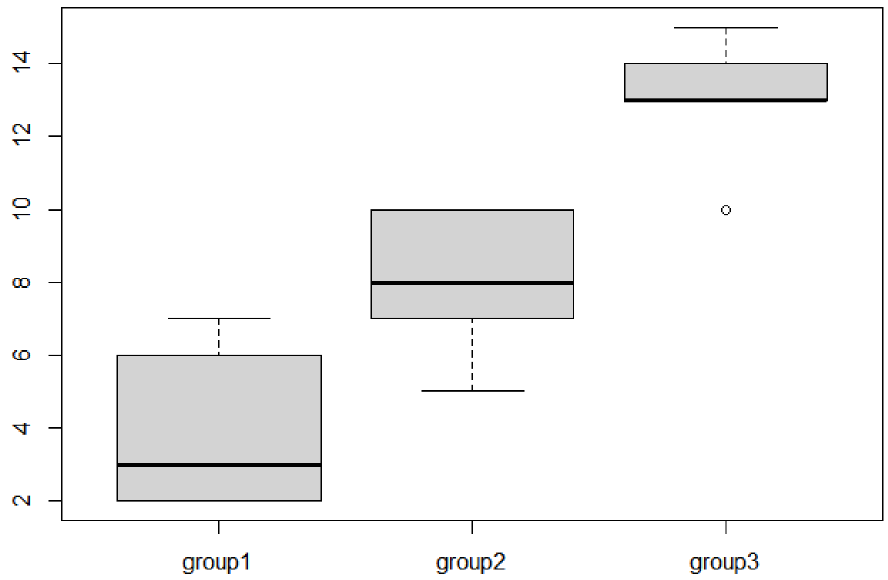
```
group1=c(2,3,7,2,6)

group2=c(10,8,7,5,10)

group3=c(10,13,14,13,15)

cg=data.frame(cbind(group1,group2,group3))

cg

boxplot(cg)

stacked_g=stack(cg)

stacked_g

av=aov(values~ind, data=stacked_g)

summary(av)

g1=c(29,30,31,31,29)

g2=c(28,29,27,30,29)

g3=c(25,28,29,27,29)

cg1=data.frame(cbind(g1,g2,g3))

cg1

stacked_g=stack(cg1)

stacked_g

av=aov(values~ind,data=stacked_g)

av1=aov(values~ind,data=stacked_g)

summary(av1)
```

Output:

```
> group1=c(2,3,7,2,6)
> group2=c(10,8,7,5,10)
> group3=c(10,13,14,13,15)
> cg=data.frame(cbind(group1,group2,group3))
> cg
  group1 group2 group3
1      2     10     10
2      3      8     13
3      7      7     14
4      2      5     13
5      6     10     15
> boxplot(cg)
> stacked_g=stack(cg)
> stacked_g
   values    ind
1       2 group1
2       3 group1
3       7 group1
4       2 group1
5       6 group1
6      10 group2
7       8 group2
8       7 group2
9       5 group2
10     10 group2
11     10 group3
12     13 group3
13     14 group3
14     13 group3
15     15 group3
```

```
> av=aov(values~ind, data=stacked_g)
> summary(av)
            Df Sum Sq Mean Sq F value   Pr(>F)    
ind          2  203.3   101.7   22.59 8.54e-05 ***
Residuals   12   54.0     4.5                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> g1=c(29,30,31,31,29)
> g2=c(28,29,27,30,29)
> g3=c(25,28,29,27,29)
> cg1=data.frame(cbind(g1,g2,g3))
> cg1
  g1 g2 g3
1 29 28 25
2 30 29 28
3 31 27 29
4 31 30 27
5 29 29 29
> stacked_g=stack(cg1)
> stacked_g
   values ind
1      29  g1
2      30  g1
3      31  g1
4      31  g1
5      29  g1
6      28  g2
7      29  g2
8      27  g2
9      30  g2
10     29  g2
11     25  g3
12     28  g3
13     29  g3
14     27  g3
15     29  g3
```

```
> av=aov(values~ind,data=stacked_g)
> av1=aov(values~ind,data=stacked_g)
> summary(av1)
            Df Sum Sq Mean Sq F value Pr(>F)  
ind          2  14.53   7.267   4.275 0.0397 *
Residuals   12  20.40   1.700                 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```



Conclusion:

We successfully performed Analysis of Variance in R.

Practical 6

Aim: Demonstration of Decision Tree

Theory:

Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.

Examples of use of decision tress is – predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these. Generally, a model is created with observed data also called training data. Then a set of validation data is used to verify and improve the model. R has packages which are used to create and visualize decision trees. For new set of predictor variable, we use this model to arrive at a decision on the category (yes/No, spam/not spam) of the data.

The R package "party" is used to create decision trees.

Use the below command in R console to install the package. You also have to install the dependent packages if any.

install.packages("party")

The package "party" has the function ctree() which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is –

ctree(formula, data)

Following is the description of the parameters used –

• formula is a formula describing the predictor and response variables.

• data is the name of the data set used.

Code:

```
x=read.csv("D:/College/Sumit/DS/weather1.csv")
x
sample_weather=sample(nrow(x),.7*nrow(x))
weather_tr=x[sample_weather,]
weather_test=x[-sample_weather,]
weather_test
library(rpart)
library(rpart.plot)
dtreemod=rpart(play.golf~.,data=weather_tr,method="class",control=rpart.control
(minsplit=1,minbucket=1))
rpart.plot(dtreemod)
p=predict(dtreemod,weather_test,type="class")
weather_test
table(weather_test$play.golf,p)
rpart.rules(dtreemod)
x2=read.csv("D:/College/Sumit/DS/weather2.csv")
x2
s2=sample(nrow(x),.7*nrow(x))
weather_tr2=x2[s2,]
weather_test2=x2[-s2,]
weather_test2
dtreemod2=rpart(Hours.Played~.,data=weather_tr2,method="anova",control=rpar
t.control(minsplit=1,minbucket=1))
rpart.plot(dtreemod2)
actuals_preds=data.frame(cbind(actuals=weather_test2$Hours.played,predicts=p))
actuals_preds
```

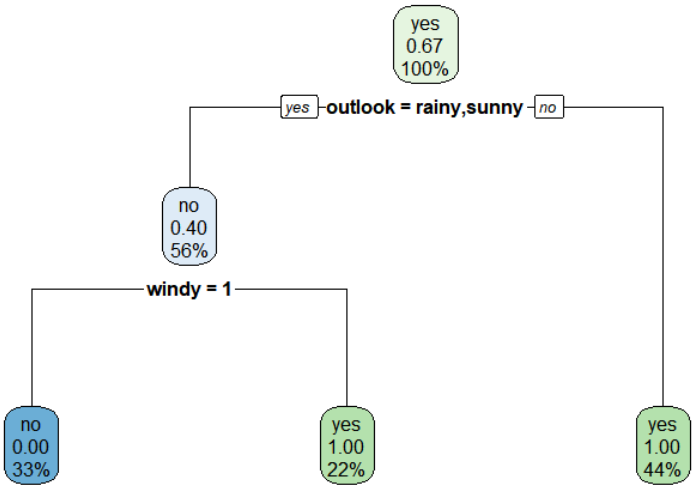
Output:

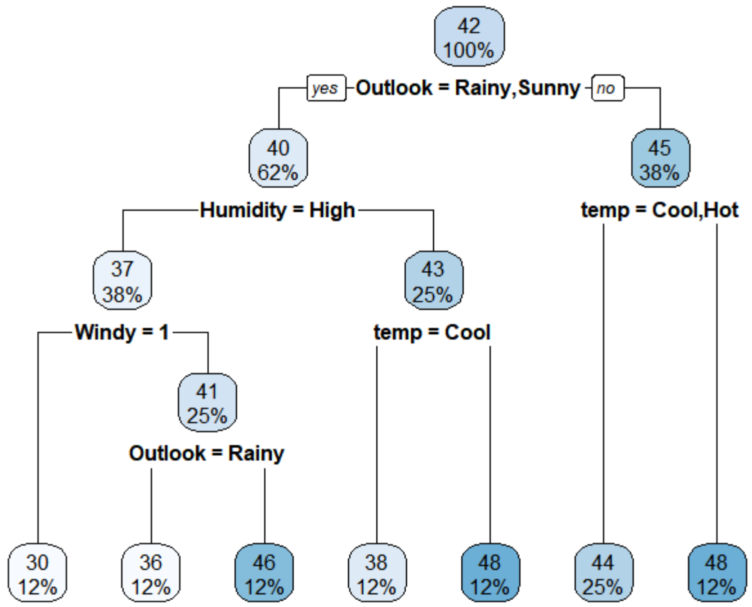
```
> x=read.csv("D:/College/Sumit/DS/weather1.csv")
> x
    outlook temp humidity windy play.golf
1     rainy  hot     high FALSE        no
2     rainy  hot     high  TRUE        no
3  overcast  hot     high FALSE       yes
4     sunny mild     high FALSE       yes
5     sunny cool   normal FALSE       yes
6     sunny cool   normal  TRUE        no
7  overcast cool   normal  TRUE       yes
8     rainy mild     high FALSE       yes
9     rainy cool   normal FALSE       yes
10    sunny mild   normal FALSE       yes
11    rainy mild   normal  TRUE       yes
12 overcast mild     high  TRUE       yes
13 overcast  hot   normal FALSE       yes
14    sunny mild     high  TRUE        no
> sample_weather=sample(nrow(x),.7*nrow(x))
> weather_tr=x[sample_weather,]
> weather_test=x[-sample_weather,]
> weather_test
   outlook temp humidity windy play.golf
1    rainy  hot     high FALSE        no
4    sunny mild     high FALSE       yes
8    rainy mild     high FALSE       yes
10   sunny mild   normal FALSE       yes
11   rainy mild   normal  TRUE       yes
> library(rpart)
> library(rpart.plot)
> dtreemod=rpart(play.golf~.,data=weather_tr,method="class",control=rpart.control(minsplit=1,minbucket=1))
> rpart.plot(dtreemod)
> p=predict(dtreemod,weather_test,type="class")
```

```
> weather_test
   outlook temp humidity windy play.golf
1    rainy  hot     high FALSE        no
4    sunny mild     high FALSE       yes
8    rainy mild     high FALSE       yes
10   sunny mild   normal FALSE       yes
11   rainy mild   normal  TRUE       yes
> table(weather_test$play.golf,p)
     p
      no yes
  no   0   1
  yes  1   3
> rpart.rules(dtreemod)
 play.golf
      0.00 when outlook is rainy or sunny & windy is 1
      1.00 when outlook is rainy or sunny & windy is 0
      1.00 when outlook is         overcast
> x2=read.csv("D:/College/Sumit/DS/weather2.csv")
> x2
    Outlook  temp Humidity Windy Hours.Played
1     Rainy  Hot     High FALSE           26
2     Rainy  Hot     High  TRUE           30
3  Overcast  Hot     High FALSE           48
4     Sunny  Mild    High FALSE           46
5     Sunny  Cool  Normal FALSE           62
6  Overcast  Cool  Normal  TRUE           43
7     Rainy  Mild    High FALSE           36
8     Rainy  Cool  Normal FALSE           38
9     Sunny  Mild  Normal FALSE           48
10    Rainy  Mild  Normal  TRUE           48
11 Overcast  Mild    High  TRUE           62
12 Overcast   Hot  Normal FALSE           44
13    Sunny  Mild    High  TRUE           30
> s2=sample(nrow(x),.7*nrow(x))
> weather_tr2=x2[s2,]
> weather_test2=x2[-s2,]
```

```
> weather_test2
    Outlook  temp Humidity Windy Hours.Played
1     Rainy  Hot     High FALSE           26
2     Rainy  Hot     High  TRUE           30
5     Sunny  Cool  Normal FALSE           62
9     Sunny  Mild  Normal FALSE           48
11 Overcast  Mild    High  TRUE           62
> dtreemod2=rpart(Hours.Played~.,data=weather_tr2,method="anova",control=rpart.control(minsplit=1,minbucket=1))
> rpart.plot(dtreemod2)
> actuals_preds=data.frame(cbind(actuals=weather_test2$Hours.played,predicts=p))
> actuals_preds
   predicts
1         2
4         2
8         2
10        2
11        1
> |
```

Charts:





Conclusion:

Thus we have successfully demonstrated the use of decision tree for playing golf.

Practical 7

Aim: Demonstration of Principal Component Analysis.

Theory:

Principal Component Analysis is one of the simple yet most powerful dimensionality reduction techniques. In simple words, PCA is a method of obtaining important variables (in the form of components) from a large set of variables available in a data set. It extracts a low-dimensional set of features by taking a projection of irrelevant dimensions from a high-dimensional data set with a motive to capture as much information as possible. With fewer variables obtained while minimizing the loss of information, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher-dimensional data.

It is always performed on a symmetric correlation or covariance matrix. This means the matrix should be numeric and have standardized data.

The covariance matrix defines the spread (variance) and the orientation (covariance) of the dataset. The direction of the spread of the dataset is computed by eigenvectors and its magnitude by eigenvalues. The no. of eigenvectors depends on the no. of principal components chosen.

Code:

```
library(FactoMineR)
x=read.csv("D:/College/Sumit/DS/student.csv")
x
cov_mat=cov(x)
cov_mat
ex=eigen(cov_mat)
ex
datapca=PCA(x,ncp=3,graph=TRUE)
datapca$eig
datapca$var
datapca$var$coord
head(iris)
x=iris[,-5]
cov_iris=cov(x)
cov_iris
irispca=PCA(x,ncp=3,graph=TRUE)
irispca summary(irispca)
```

Output:

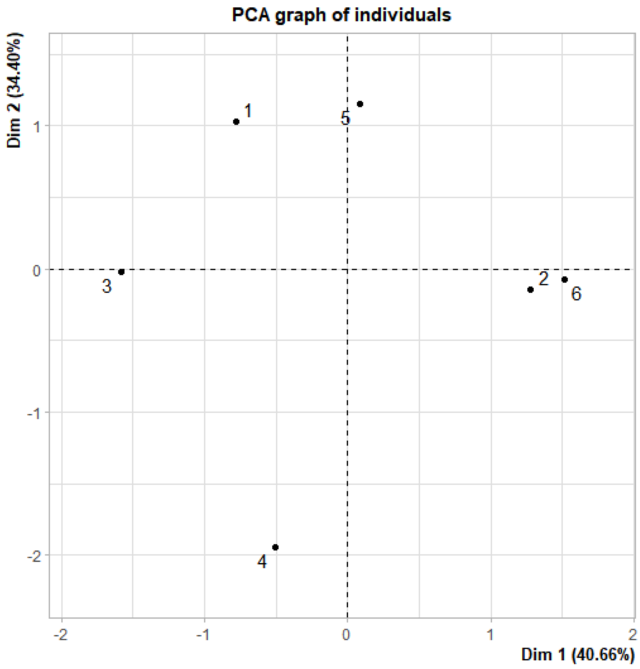
```
> library(FactoMineR)
> x=read.csv("D:/College/Sumit/DS/student.csv")
> x
  Maths English Arts
1    90      80   60
2    67      73   88
3    92      66   50
4    77      49   69
5    55      73   40
6    59      72   84
> cov_mat=cov(x)
> cov_mat
            Maths    English       Arts
Maths   244.26667 -17.533333 -61.666667
English -17.53333 114.166667  -8.166667
Arts    -61.66667  -8.166667 356.166667
> ex=eigen(cov_mat)
> ex
eigen() decomposition
$values
[1] 383.4829 220.4690 110.6481

$vectors
             [,1]       [,2]        [,3]
[1,]  0.404857141  0.8997141 -0.16311118
[2,]  0.001369887 -0.1789811 -0.98385156
[3,] -0.914378925  0.3980959 -0.07369427

> datapca=PCA(x,ncp=3,graph=TRUE)
> datapca$eig
       eigenvalue percentage of variance
comp 1  1.2197327               40.65776
comp 2  1.0321276               34.40425
comp 3  0.7481397               24.93799
       cumulative percentage of variance
comp 1                          40.65776
comp 2                          75.06201
comp 3                         100.00000
```

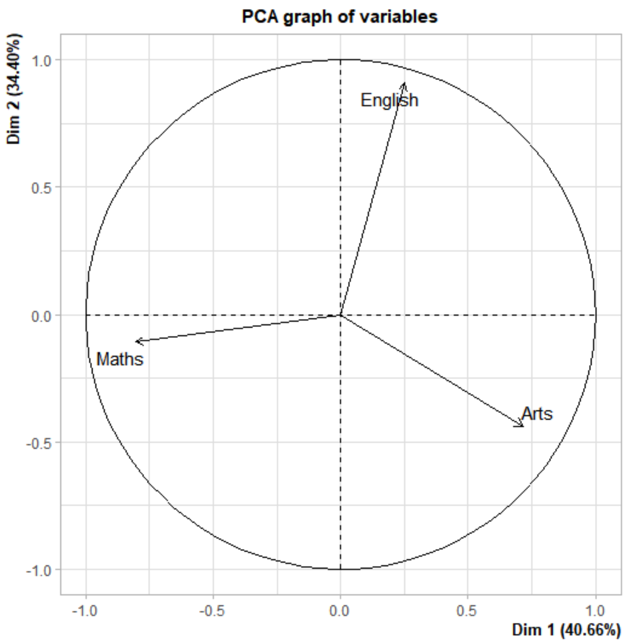
```
> datapca$var
$coord
             Dim.1      Dim.2     Dim.3
Maths   -0.8018925 -0.1085306 0.5875284
English  0.2510661  0.9092786 0.3319311
Arts     0.7167056 -0.4399559 0.5410840

$cor
             Dim.1      Dim.2     Dim.3
Maths   -0.8018925 -0.1085306 0.5875284
English  0.2510661  0.9092786 0.3319311
Arts     0.7167056 -0.4399559 0.5410840

$cos2
             Dim.1      Dim.2     Dim.3
Maths   0.64303154 0.01177889 0.3451896
English 0.06303416 0.82678756 0.1101783
Arts    0.51366697 0.19356118 0.2927719

$contrib
            Dim.1     Dim.2    Dim.3
Maths   52.719055  1.141224 46.13972
English  5.167867 80.105167 14.72697
Arts    42.113078 18.753609 39.13331

> datapca$var$coord
             Dim.1      Dim.2     Dim.3
Maths   -0.8018925 -0.1085306 0.5875284
English  0.2510661  0.9092786 0.3319311
Arts     0.7167056 -0.4399559 0.5410840
```

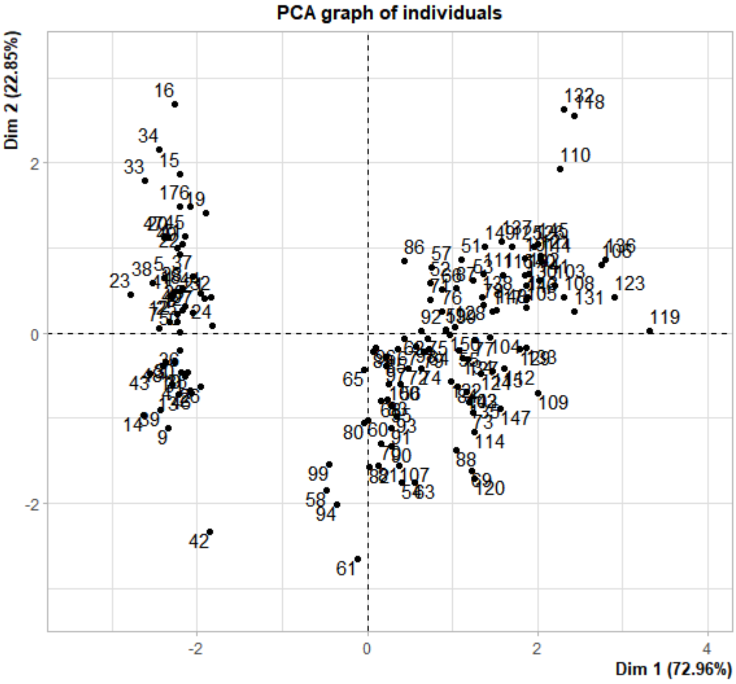
```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> x=iris[,-5]
> cov_iris=cov(x)
> cov_iris
             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
> irispca=PCA(x,ncp=3,graph=TRUE)
> irispca
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 150 individuals, described by 4 variables
*The results are available in the following objects:

   name               description
1  "$eig"             "eigenvalues"
2  "$var"             "results for the variables"
3  "$var$coord"       "coord. for the variables"
4  "$var$cor"         "correlations variables - dimensions"
5  "$var$cos2"        "cos2 for the variables"
6  "$var$contrib"     "contributions of the variables"
7  "$ind"             "results for the individuals"
8  "$ind$coord"       "coord. for the individuals"
9  "$ind$cos2"        "cos2 for the individuals"
10 "$ind$contrib"     "contributions of the individuals"
11 "$call"            "summary statistics"
```

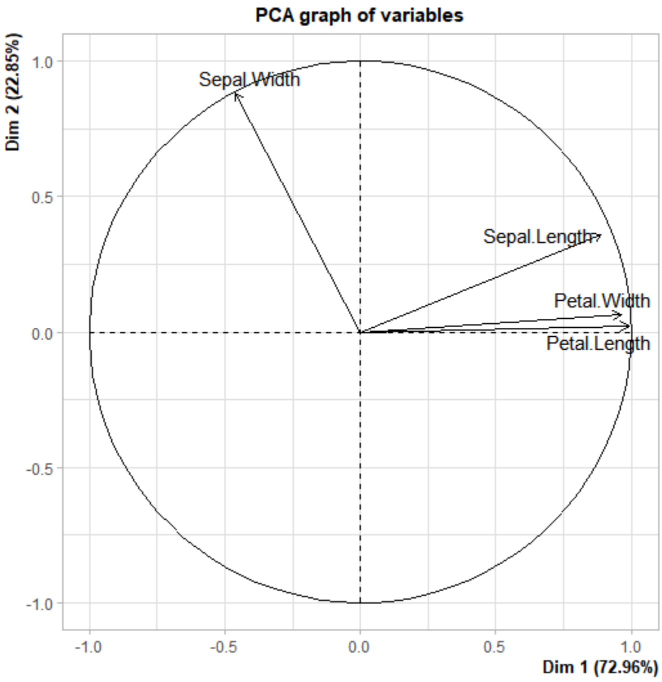
```
12 "$call$centre"      "mean of the variables"
13 "$call$ecart.type"  "standard error of the variables"
14 "$call$row.w"       "weights for the individuals"
15 "$call$col.w"       "weights for the variables"
> summary(irispca)

Call:
PCA(X = x, ncp = 3, graph = TRUE)


Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4
Variance               2.918   0.914   0.147   0.021
% of var.             72.962  22.851   3.669   0.518
Cumulative % of var.  72.962  95.813  99.482 100.000

Individuals (the 10 first)
               Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2
1          |  2.319 | -2.265  1.172  0.954 |  0.480  0.168  0.043
2          |  2.202 | -2.081  0.989  0.893 | -0.674  0.331  0.094
3          |  2.389 | -2.364  1.277  0.979 | -0.342  0.085  0.020
4          |  2.378 | -2.299  1.208  0.935 | -0.597  0.260  0.063
5          |  2.476 | -2.390  1.305  0.932 |  0.647  0.305  0.068
6          |  2.555 | -2.076  0.984  0.660 |  1.489  1.617  0.340
7          |  2.468 | -2.444  1.364  0.981 |  0.048  0.002  0.000
8          |  2.246 | -2.233  1.139  0.988 |  0.223  0.036  0.010
9          |  2.592 | -2.335  1.245  0.812 | -1.115  0.907  0.185
10         |  2.249 | -2.184  1.090  0.943 | -0.469  0.160  0.043
              Dim.3    ctr   cos2
1          | -0.128  0.074  0.003 |
2          | -0.235  0.250  0.011 |
3          |  0.044  0.009  0.000 |
4          |  0.091  0.038  0.001 |
5          |  0.016  0.001  0.000 |
6          |  0.027  0.003  0.000 |
7          |  0.335  0.511  0.018 |
```

Graphs:


PCA graph of individuals
Dim 2 (34.40%)
Dim 1 (40.66%)
1
2
3
4
5
6
1
0
-1
-2
-2
-1
0
1
2


PCA graph of variables
Dim 2 (34.40%)
Dim 1 (40.66%)
English
Maths
Arts
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0





Conclusion:

Thus successfully demonstrated the use of Principal Component Analysis for a given data set.

Practical 8

Aim: Demonstration of Clustering

Theory:

Clustering is the most widespread and popular method of Data Analysis and Data Mining. It used in cases where the underlying input data has a colossal volume and we are tasked with finding similar subsets that can be analysed in several ways.

For example – A marketing company can categorise their customers based on their economic background, age and several other factors to sell their products, in a better way.

Applications of R clustering are as follows:

• Marketing – In the area of marketing, we use clustering to explore and select customers that are potential buyers of the product. This differentiates the most likeable customers from the ones who possess the least tendency to purchase the product. After the clusters have been developed, businesses can keep a track of their customers and make necessary decisions to retain them in that cluster.

• Retail – Retail industries make use of clustering to group customers based on their preferences, style, choice of wear as well as store preferences. This allows them to manage their stores in a much more efficient manner.

• Medical Science – Medicine and health industries make use of clustering algorithms to facilitate efficient diagnosis and treatment of their patients as well as the discovery of new medicines. Based on the age, group, genetic coding of the patients, these organisations are better capable to understand diagnosis through robust clustering.

• Sociology – Clustering is used in Data Mining operations to divide people based on their demographics, lifestyle, socioeconomic status, etc. This can help the law enforcement agencies to group potential criminals and even identify them with an efficient implementation of the clustering algorithm.

Code:

```
"K-means Clustering"

data(iris)

names(iris)

new_data<-subset(iris,select = c(-Species))

new_data

cl<-kmeans(new_data,3)

cl

data<-new_data

wss<-sapply(1:15,function(k){kmeans(data,k)$tot.withinss})

wss

plot(1:15,wss,type="b",pch=19,frame=FALSE,xlab   ="Number   ofclusters K",ylab =
"Total within-clusters sums of squares")

library(cluster)

clusplot(new_data,cl$cluster,color=TRUE,shade=TRUE,labels=2,lines=0)

cl$cluster

cl$centers

"agglomarative clustering"

clusters<-hclust(dist(iris[,3:4]))

plot(clusters)

clusterCut<-cutree(clusters,3)

table(clusterCut,iris$Species)
```

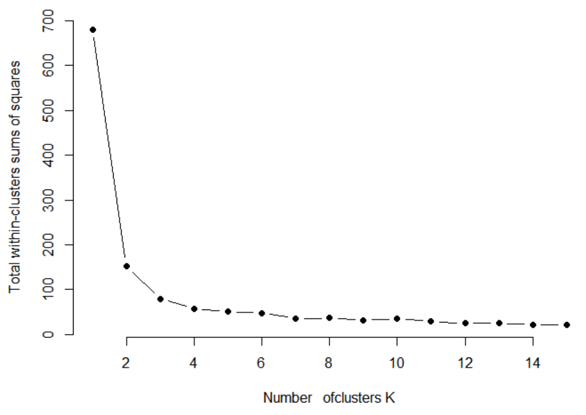
Output:

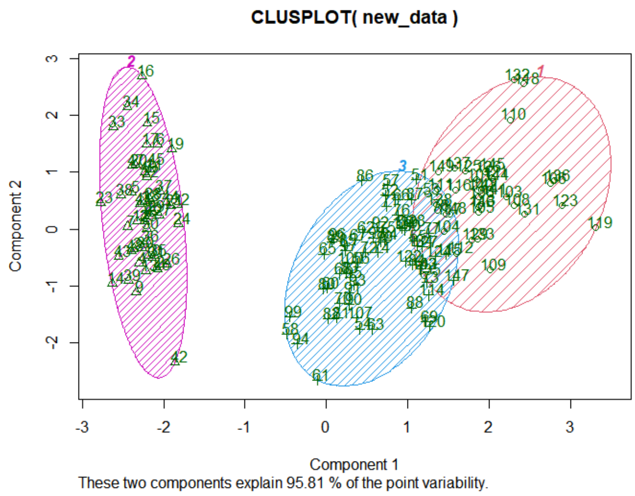
```
> "K-means Clustering"
[1] "K-means Clustering"
> data(iris)
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> new_data<-subset(iris,select = c(-Species))
> new_data
    Sepal.Length Sepal.Width Petal.Length Petal.Width
1            5.1         3.5          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
```

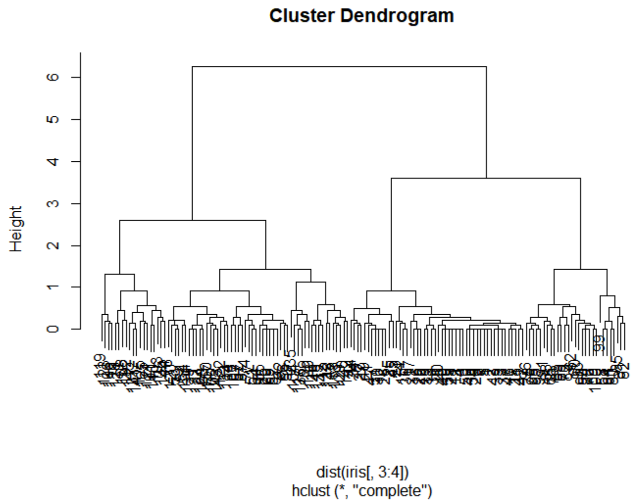
```
> cl<-kmeans(new_data,3)
> cl
K-means clustering with 3 clusters of sizes 50, 38, 62

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

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
 37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3 
 73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 
  3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   2   2   2   2   3   2 
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 
  2   2   2   2   2   3   3   2   2   2   2   3   2   3   2   3   2   2   3   3   2   2   2   2   2   3   2   2   2   2   3   2   2   2   3   2 
145 146 147 148 149 150 
  2   2   3   2   2   3 

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

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"      
> data<-new_data
> wss<-sapply(1:15,function(k){kmeans(data,k)$tot.withinss})
> wss
 [1] 681.37060 152.34795  78.85144  57.26562  49.85942  45.51845  34.29823  35.98358  33.95628  27.34710  25.77609  29.03927  23.00880  22.52904
[15]  21.23357
```

```
> plot(1:15,wss,type="b",pch=19,frame=FALSE,xlab   ="Number   ofclusters K",ylab = "Total within-clusters sums of squares")
> library(cluster)
> clusplot(new_data,cl$cluster,color=TRUE,shade=TRUE,labels=2,lines=0)
> cl$cluster
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
 37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3 
 73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 
  3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   2   2   2   2   3   2 
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 
  2   2   2   2   2   3   3   2   2   2   2   3   2   3   2   3   2   2   3   3   2   2   2   2   2   3   2   2   2   2   3   2   2   2   3   2 
145 146 147 148 149 150 
  2   2   3   2   2   3 
> cl$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     6.850000    3.073684     5.742105    2.071053
3     5.901613    2.748387     4.393548    1.433871
> "agglomarative clustering"
[1] "agglomarative clustering"
> clusters<-hclust(dist(iris[,3:4]))
> plot(clusters)
> clusterCut<-cutree(clusters,3)
> table(clusterCut,iris$Species)
          
clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         21        50
         3      0         29         0
> |
```

Graphs:







Conclusion:

Thus we have successfully learnt and demonstrated the use of clustering in R with the help of a dataset.

Practical 9

Aim: Demonstration of Time-series forecasting.

Thoery :

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

The time series object is created by using the ts() function.

Syntax

The basic syntax for ts() function in time series analysis is –

timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used –

- data is a vector or matrix containing the values used in the time series.
- start specifies the start time for the first observation in time series.
- end specifies the end time for the last observation in time series.
- frequency specifies the number of observations per unit time.

Except the parameter "data" all other parameters are optional.

A time series can be broken down to its components so as to systematically understand, analyze, model and forecast it.

A time series with additive trend, seasonal, and irregular components can be decomposed using the stl() function. Note that a series with multiplicative effects can often by transformed into series with additive effects through a log transformation (i.e., newts <- log(myts)).

Code:

```
install.packages("timeSeries")

install.packages("forecast")

library(timeSeries)

library(forecast)

x=table(AirPassengers)

x

View(x)

frequency(AirPassengers)

tsdata=ts(AirPassengers,frequency=12)

tsdata

plot(tsdata)

d=decompose(tsdata,"multiplicative")

plot(d)

plot(d$trend)

plot(d$random)

boxplot(AirPassengers~cycle(AirPassengers,xlab="date",ylab="passengers count in 1000",main="monthly box plot"))

mymodel<- arima(AirPassengers)

mymodel
```

Output:

```
> install.packages("timeSeries")
WARNING: Rtools is required to build R packages but is not currently installed. Please downlo
opriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
also installing the dependency 'timeDate'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/timeDate_4022.108.zip'
Content type 'application/zip' length 1378059 bytes (1.3 MB)
downloaded 1.3 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/timeSeries_4021.105.zip'
Content type 'application/zip' length 2019046 bytes (1.9 MB)
downloaded 1.9 MB

package 'timeDate' successfully unpacked and MD5 sums checked
package 'timeSeries' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pc\AppData\Local\Temp\Rtmp2zHXXt\downloaded_packages
```

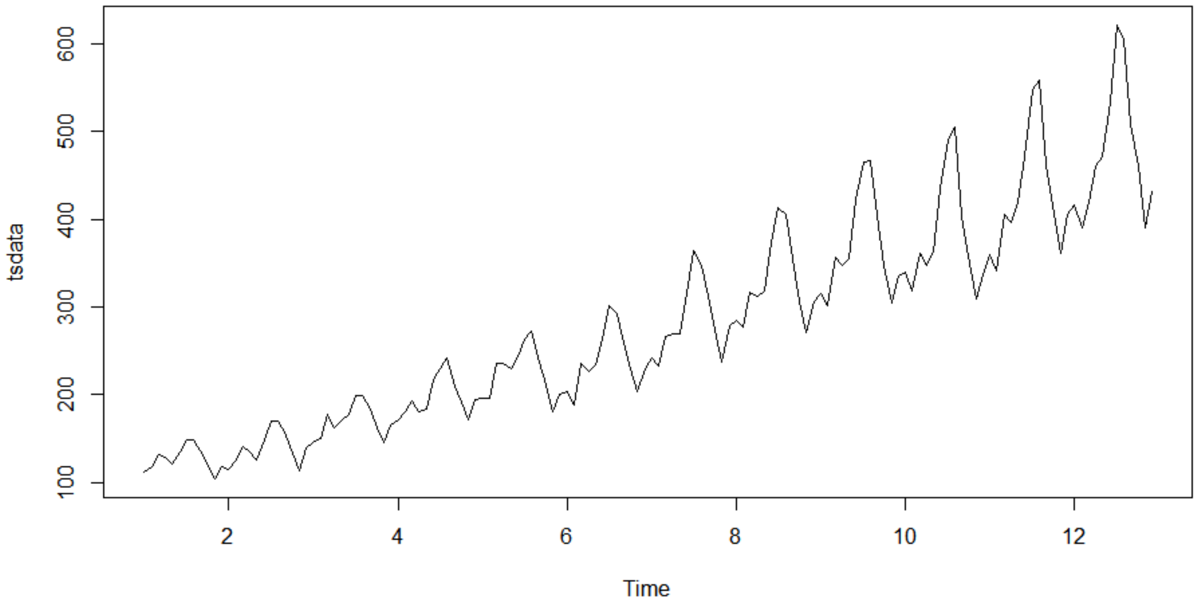
```
> install.packages("forecast")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appr
opriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
also installing the dependencies 'xts', 'TTR', 'quadprog', 'quantmod', 'fracdiff', 'generics', 'lmtest', 'tseries',
 'urca', 'zoo', 'RcppArmadillo'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/xts_0.13.0.zip'
Content type 'application/zip' length 903220 bytes (882 KB)
downloaded 882 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/TTR_0.24.3.zip'
Content type 'application/zip' length 535953 bytes (523 KB)
downloaded 523 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/quadprog_1.5-8.zip'
Content type 'application/zip' length 50391 bytes (49 KB)
downloaded 49 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/quantmod_0.4.20.zip'
Content type 'application/zip' length 1037440 bytes (1013 KB)
downloaded 1013 KB
```

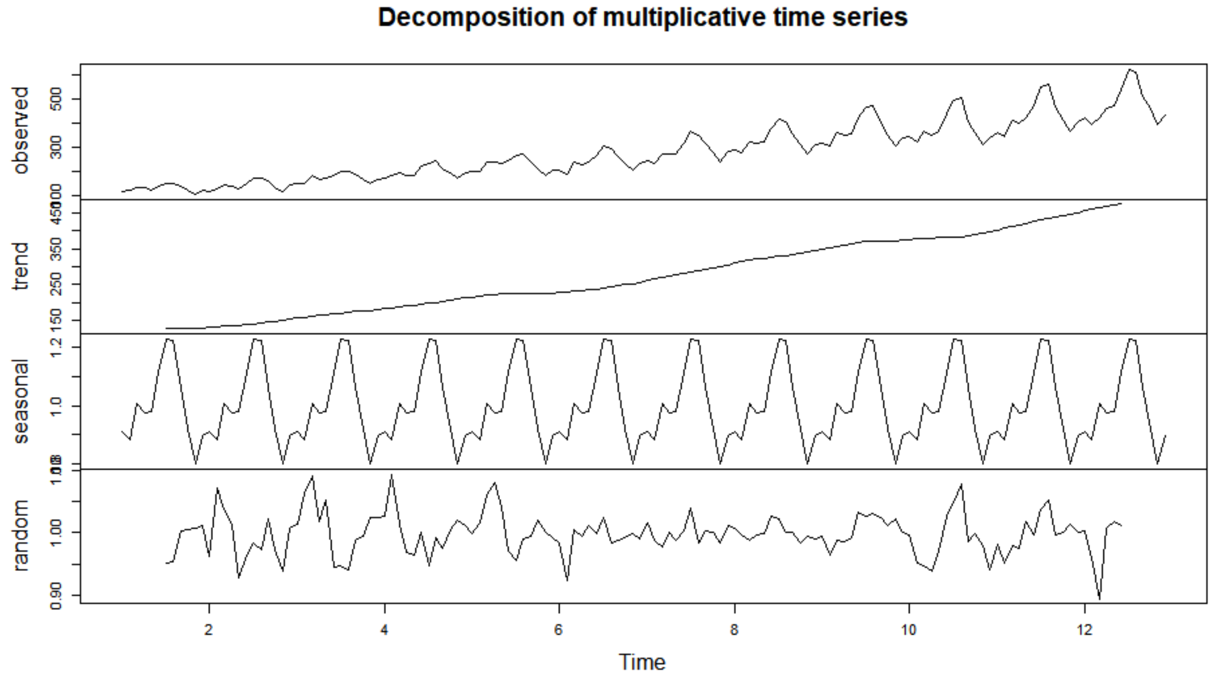
```
> library(timeSeries)
Loading required package: timeDate
Warning messages:
1: package 'timeSeries' was built under R version 4.1.3
2: package 'timeDate' was built under R version 4.1.3
> library(forecast)
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
Warning message:
package 'forecast' was built under R version 4.1.3
> x=table(AirPassengers)
> x
AirPassengers
104 112 114 115 118 119 121 125 126 129 132 133 135 136 140 141 145 146 148 149 150 158 162 163 166 170 171 172
  1   1   1   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1   2   1   2
178 180 181 183 184 188 191 193 194 196 199 201 203 204 209 211 218 227 229 230 233 234 235 236 237 242 243 259
  2   2   1   1   1   1   1   1   1   2   2   1   1   1   1   1   1   1   3   1   1   1   2   1   2   2   1   1
264 267 269 270 271 272 274 277 278 284 293 301 302 305 306 310 312 313 315 317 318 336 337 340 342 347 348 355
  2   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   2   1   2   1   1   1   1   2   2   2
356 359 360 362 363 364 374 390 391 396 404 405 406 407 413 417 419 420 422 432 435 461 463 465 467 472 491 505
  1   1   1   2   1   1   1   1   1   1   2   2   1   1   1   1   1   1   1   1   1   2   1   1   1   2   1   1
508 535 548 559 606 622
  1   1   1   1   1   1
> View(x)
> frequency(AirPassengers)
[1] 12
> tsdata=ts(AirPassengers,frequency=12)
> tsdata
```

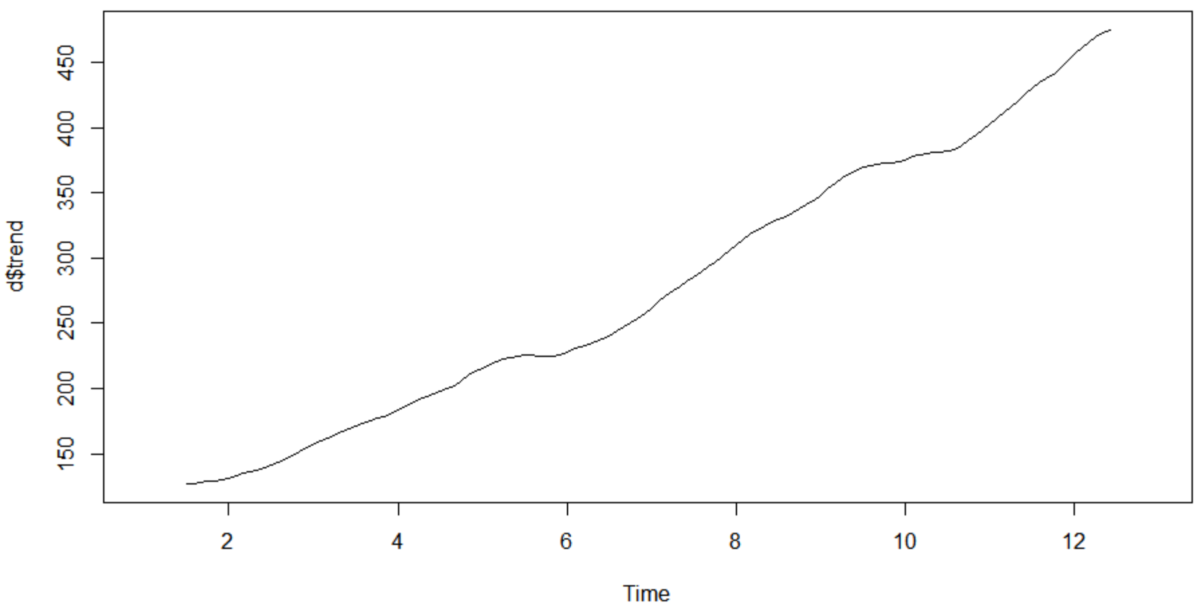
```
> tsdata=ts(AirPassengers,frequency=12)
> tsdata
   Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1  112 118 132 129 121 135 148 148 136 119 104 118
2  115 126 141 135 125 149 170 170 158 133 114 140
3  145 150 178 163 172 178 199 199 184 162 146 166
4  171 180 193 181 183 218 230 242 209 191 172 194
5  196 196 236 235 229 243 264 272 237 211 180 201
6  204 188 235 227 234 264 302 293 259 229 203 229
7  242 233 267 269 270 315 364 347 312 274 237 278
8  284 277 317 313 318 374 413 405 355 306 271 306
9  315 301 356 348 355 422 465 467 404 347 305 336
10 340 318 362 348 363 435 491 505 404 359 310 337
11 360 342 406 396 420 472 548 559 463 407 362 405
12 417 391 419 461 472 535 622 606 508 461 390 432
> plot(tsdata)
> d=decompose(tsdata,"multiplicative")
> plot(d)
> plot(d$trend)
> plot(d$random)
> boxplot(AirPassengers~cycle(AirPassengers,xlab="date",ylab="passengers count in 1000",main="monthly box plot"))
> mymodel<- arima(AirPassengers)
> mymodel

Call:
arima(x = AirPassengers)

Coefficients:
      intercept
       280.2986
```

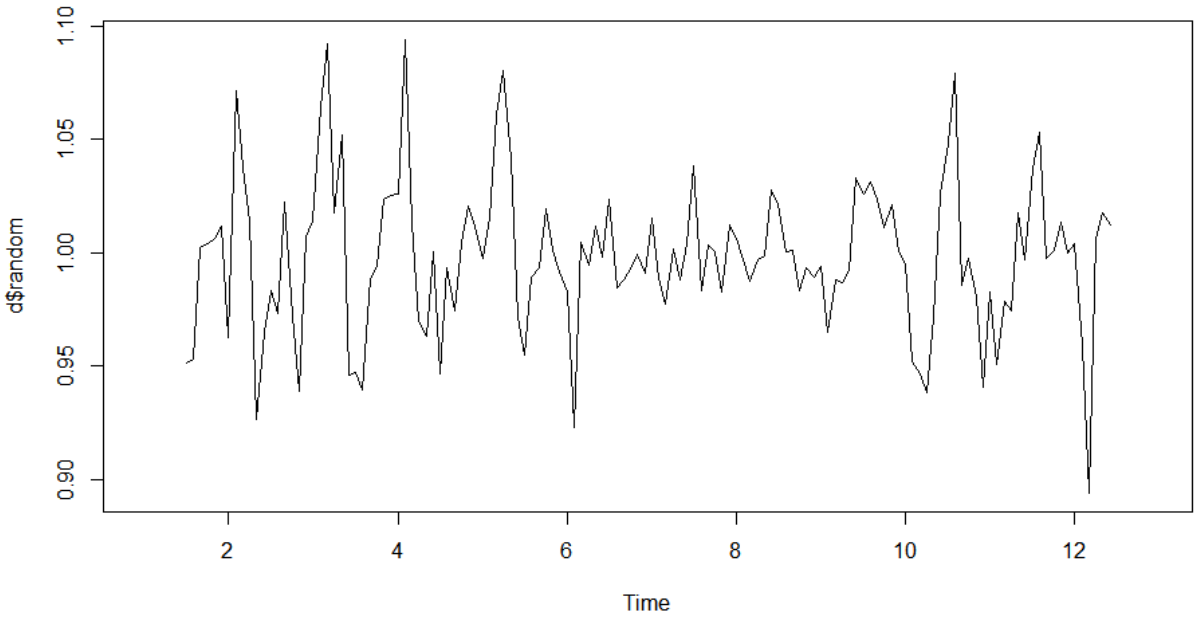
```
Coefficients:
      intercept
       280.2986
s.e.     9.9624

sigma^2 estimated as 14292:  log likelihood = -893.18,  aic = 1790.37
>
```

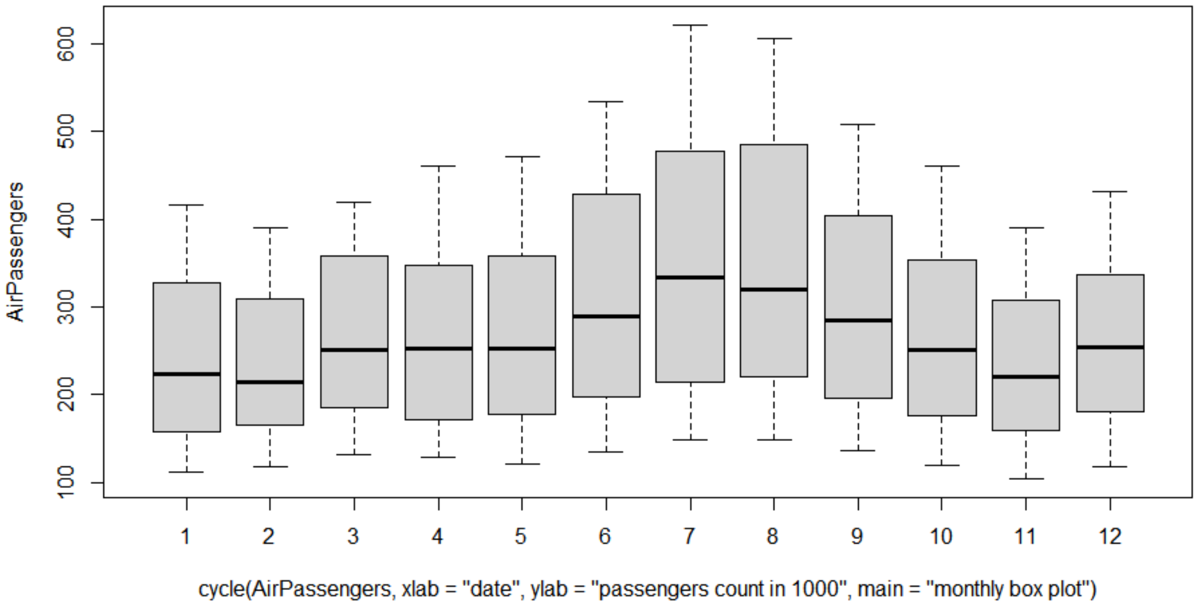
Graphs:




Decomposition of multiplicative time series
observed
trend
seasonal
random
Time


d$trend
Time





Conclusion:

Thus we have successfully understood and demonstrated the use of time series forecasting with a data set.

Practical 10

Aim: Use of R Markdown and RStudio Cloud (Store mini project in RStudio Cloud)

Theory:

Clustering is the most widespread and popular method of Data Analysis and Data Mining. It used in cases where the underlying input data has a colossal volume and we are tasked with finding similar subsets that can be analysed in several ways.

For example – A marketing company can categorise their customers based on their economic background, age and several other factors to sell their products, in a better way.

Applications of R clustering are as follows:

- Marketing – In the area of marketing, we use clustering to explore and select customers that are potential buyers of the product. This differentiates the most likeable customers from the ones who possess the least tendency to purchase the product. After the clusters have been developed, businesses can keep a track of their customers and make necessary decisions to retain them in that cluster.
- Retail – Retail industries make use of clustering to group customers based on their preferences, style, choice of wear as well as store preferences. This allows them to manage their stores in a much more efficient manner.
- Medical Science – Medicine and health industries make use of clustering algorithms to facilitate efficient diagnosis and treatment of their patients as well as the discovery of new medicines. Based on the age, group, genetic coding of the patients, these organisations are better capable to understand diagnosis through robust clustering.
- Sociology – Clustering is used in Data Mining operations to divide people based on their demographics, lifestyle, socioeconomic status, etc. This can help the law enforcement agencies to group potential criminals and even identify them with an efficient implementation of the clustering algorithm.

Code:

```
df=read.csv("AGE.csv")

df

plot(df)

boxplot(df)

c1=kmeans(df[1:2],3)

c1

iris

View(iris)

head(iris)

plot(iris)

plot(iris[,3:4])

kmeansc1=kmeans(iris[,3:4],3)

kmeansc1

table(kmeansc1$cluster,iris$ Species)

boxplot(iris)
```

Output:

```
  could not find function "Table"
> df=read.csv("AGE.csv")
> df
   AGE SPEND
1   18    10
2   20    25
3   22    30
4   24    10
5   26    25
6   28    30
7   30    80
8   32    14
9   34    45
10  36    78
11  38    45
12  40    56
13  42     5
14  44    56
15  46    56
16  48     0
17  50    55
18  52    89
19  54    55
20  56    56
> plot(df)
> boxplot(df)
> c1=kmeans(df[1:2],3)
> c1
K-means clustering with 3 clusters of sizes 5, 9, 6

Cluster means:
       AGE    SPEND
1 32.80000  7.80000
2 45.33333 64.55556
```

```
2 45.33333 64.55556
3 28.00000 33.33333

Clustering vector:
 [1] 1 3 3 1 3 3 2 1 3 2 3 2 1 2 2 1 2 2 2 2

Within cluster sum of squares by cluster:
[1]  729.6000 2100.2222  673.3333
 (between_SS / total_SS =  77.5 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> 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
```

```
150  virginica
> View(iris)
> head(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
> plot(iris)
> plot(iris[,3:4])
> kmeansc1=kmeans(iris[,3:4],3)
> kmeansc1
K-means clustering with 3 clusters of sizes 50, 46, 54

Cluster means:
  Petal.Length Petal.Width
1     1.462000    0.246000
2     5.626087    2.047826
3     4.292593    1.359259

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [28] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3
 [55] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
 [82] 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 2
[109] 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 3 2 2 3 3 2 2 2 2 2 2 2
[136] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1]  2.02200 15.16348 14.22741
```

```
Within cluster sum of squares by cluster:
[1]  2.02200 15.16348 14.22741
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> table(kmeansc1$cluster,iris$ Species)

    setosa versicolor virginica
  1     50          0         0
  2      0          2        44
  3      0         48         6
> boxplot(iris)
>
```

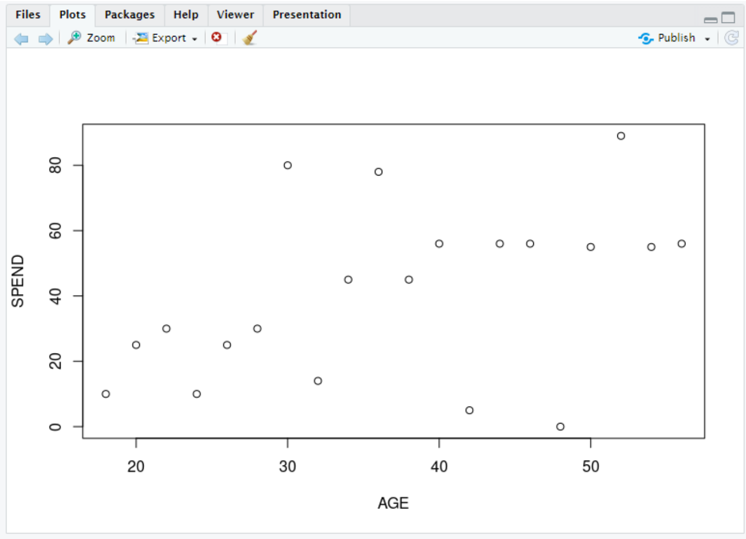
Untitled1* × | iris ×

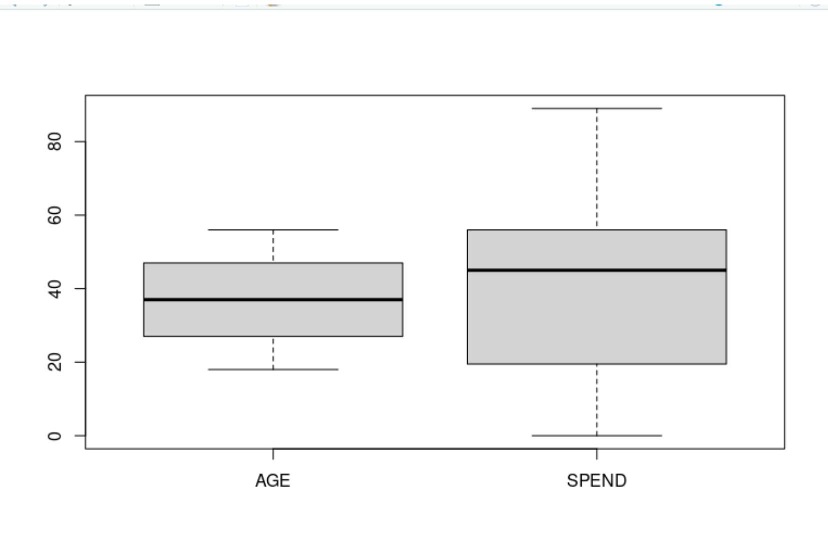
Filter

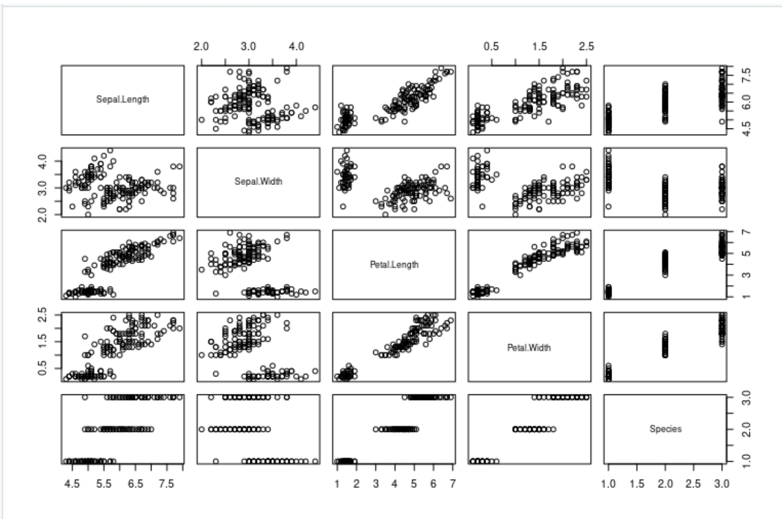
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa

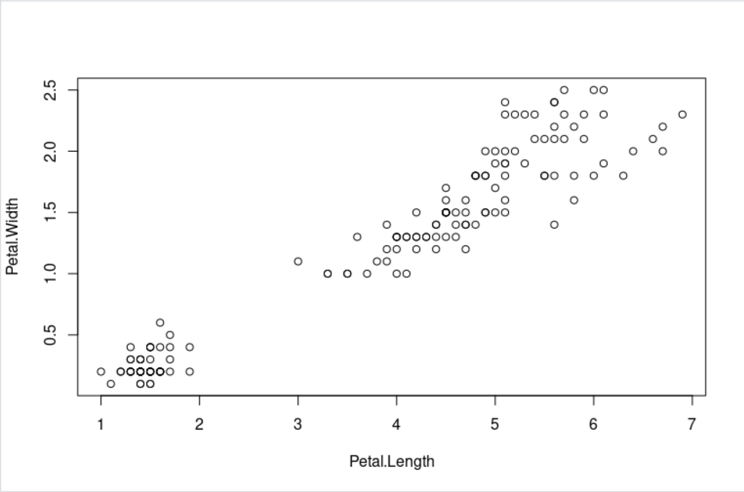
Showing 1 to 11 of 150 entries, 5 total columns

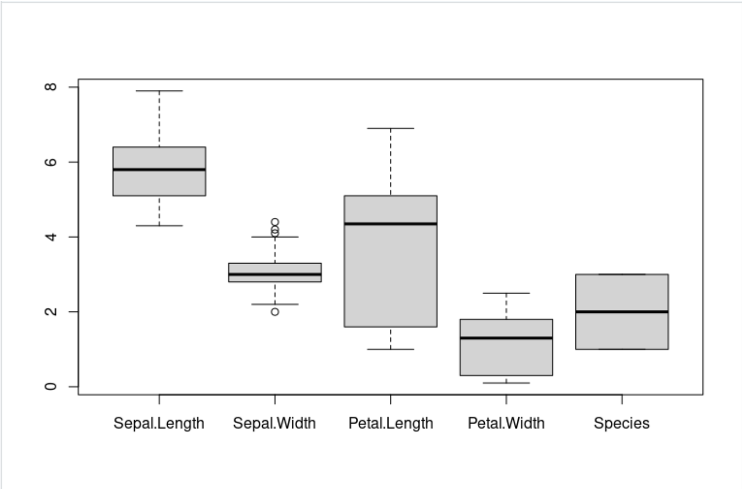
Graphs:


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
SPEND
0
20
40
60
80
20
30
40
50
AGE


0
20
40
60
80
AGE
SPEND


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
2.0
3.0
4.0
0.5
1.5
2.5
4.5
6.0
7.5
1
3
5
7
1.0
2.0
3.0
4.5
5.5
6.5
7.5
1
2
3
4
5
6
7
1.0
1.5
2.0
2.5
3.0





Conclusion:

Thus we performed kmeans clustering on r cloud