UNIVERSAL

LORDS UNIVERSAL COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

2022-2023

Certificate

This is to certify that Mr/Miss. **Gautami Anand Rathwad** with Roll No. **23**

Has successfully completed the necessary Course of experiments in the subject

of **Data Science** during the Academic year **2022-2023** complying with the

Requirements of **University of Mumbai** for the program **TY BSc. (COMPUTER SCIENCE)**

in Semester VI.

Internal Examiner	External Examiner
Co-Ordinator	Stamp

R is an open-source programming language that facilitates statistical computing and graphical libraries. Being open-source, R enjoys community support of avid developers who work on releasing new packages, updating R and making it a steadfast programming package for Data Science.

- With the help of R, one can perform various statistical operations.
- You can obtain it for free from the website www.r-project.org.
- It is driven by command lines.
- Each command is executed when the user enters them into the prompt.

Since R is open-source, most of its routines and procedures have been developed by programmers all over the world. All the packages are available for free at the R project website called CRAN. It contains over 10,000 packages in R. The basic installation comprises of a set of tools that various data scientists and statisticians use for multiple tasks.
In R, there is a comprehensive environment that facilitates the performance of statistical operations as well as the generation of data analysis in graphical or text format. The commands that a console takes in as input are assessed and subsequently executed. R is incapable of handling auto-formatting characters such as dashes or quotes, hence, you need to be discreet while copy-pasting commands from external sources into your R environment.

- R is a comprehensive programming language that provides support for procedural programming involving functions as well as object-oriented programming with generic functions.
- There are more than 10,000 packages in the repository of R programming. With these packages, one can make use of functions to facilitate easier programming.
- Being an interpreter based language, R produces a machine-independent code that is portable in nature. Furthermore, it facilitates easy debugging of errors in the code.
- R facilitates complex operations with vectors, arrays, data frames as well as other data objects that have varying sizes.
- R can be easily integrated with many other technologies and frameworks like Hadoop and HDFS. It can also integrate with other programming languages like C, C++, Python, Java, FORTRAN, and JavaScript.
- R provides robust facilities for data handling and storage.
- As discussed in the above section, R has extensive community support that provides technical assistance, seminars and several boot camps to get you started with R.
- R is cross-platform compatible. R packages can be installed and used on any OS in any software environment without any changes.

There are certain unique aspects of R programming which makes it better in comparison with other technologies:

- **Graphical Libraries –** R stays ahead of the curve through its aesthetic graphical libraries. Libraries like ggplot2, plotly facilitate appealing libraries for making well-defined plots.
- Availability / Cost – R is completely free to use which means widespread availability.
- Advancement in Tool – R supports various advanced tools and features that allow you to build robust statistical models.
- Job Scenario – As stated above, R is the primary tool for Data Science. With the immense growth in Data Science and rise in demand, R has become the most in-demand programming language of the world today.
- Customer Service Support and Community – With R, you can enjoy strong community support.
- Portability – R is highly portable. Many different programming languages and software frameworks can easily combine with the R environment for the best results.

Applications of R Programming

- R is used in finance and banking sectors for detecting fraud, reducing customer churn rate and for making future decisions.
- R is also used by bioinformatics to analyze strands of genetic sequences, for performing drug discovery and also in computational neuroscience.
- R is used in social media analysis to discover potential customers in online advertising. Companies also use social media information to analyses customer sentiments for making their products better.
- E-Commerce companies make use of R to analyze the purchases made by the customers as well as their feedbacks.
- Manufacturing companies use R to analyze customer feedback. They also use it to predict future demand to adjust their manufacturing speeds and maximize profits.

Practical 1

Aim: Practical of Data collection, Data curation and management for unstructured data (NoSQL)

Description

Data Collection**:**

Data collection is the process of collecting, measuring, and analysing data from various sources to gain insights. Data can be collected through various sources, such as social media monitoring, online tracking, surveys, feedback, etc. In fact, there are three main categories of data that businesses endeavour to collect.

Data Curation:

Data curation is the process of creating, organizing and maintaining data sets so they can be accessed and used by people looking for information. It involves collecting, structuring, indexing and cataloging data for users in an organization, group or the general public. Data can be curated to support business decision-making, academic needs, scientific research and other purposes.

Data curation is part of the overall data management process and sometimes is incorporated into data preparation work that gets data sets ready for use in business intelligence (BI) and analytics applications. In other cases, prepared data may be fed into the curation process for ongoing management and maintenance. Some organizations have formal data curator positions -- in ones that don't, data stewards, data engineers, database administrators, data scientists or business users may fill that role.

Dplyr:

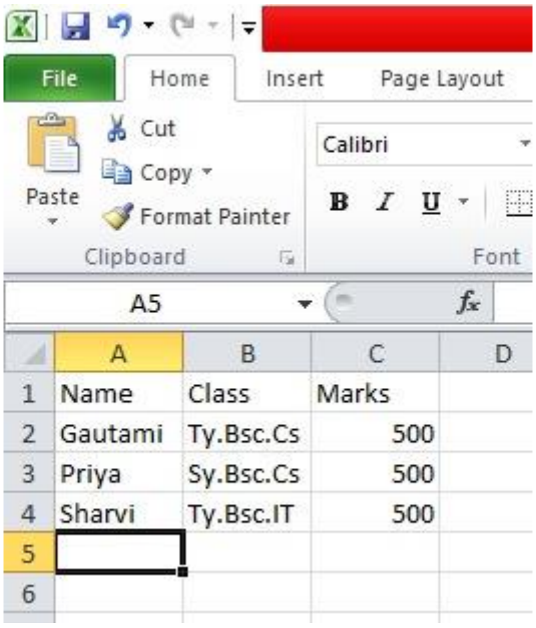
The dplyr package in R Programming Language is a structure of data manipulation that provides a uniform set of verbs, helping to resolve the most frequent data manipulation hurdles.

Code:

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

names(stud)

dim(stud)

str(stud)

query1<-filter(stud,Marks>750|Class=="TyCs")

query1
```

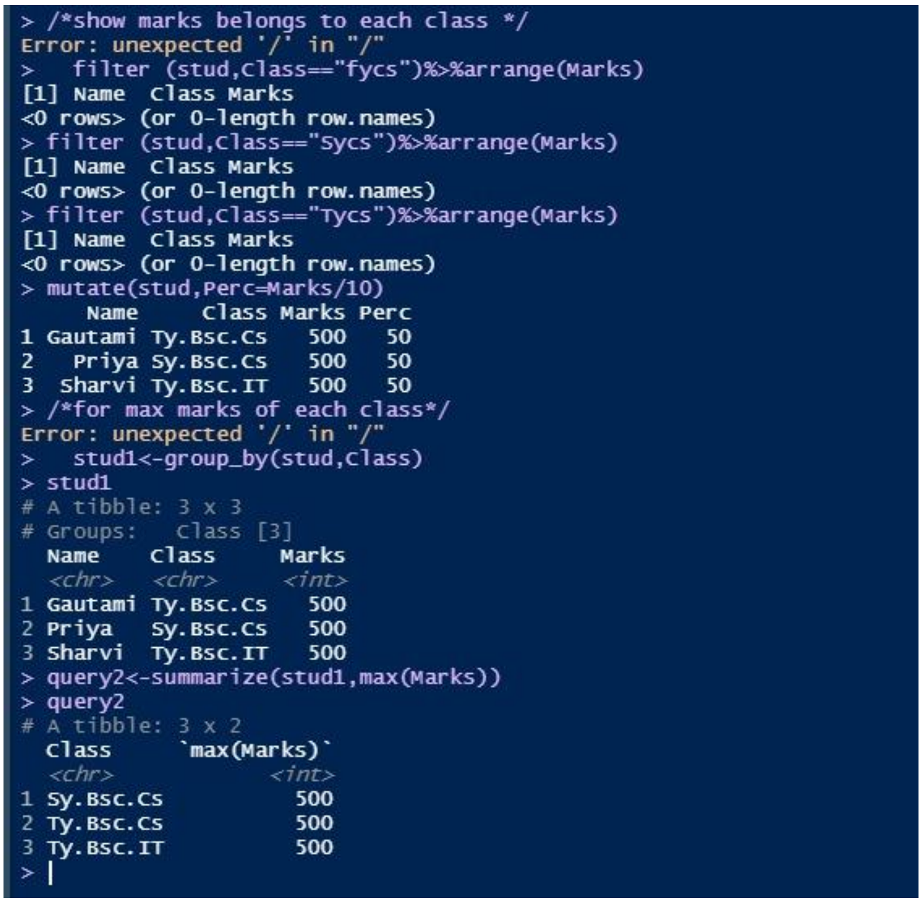
```
stud1<-group_by(stud,Class)

stud1

query2<-summarise(stud1,mean(Marks))

query2

summarise(stud,max(Marks))

summarise(stud,min(Marks))

summarise(stud,n())

summarise(stud,sum(Marks))

query3<-arrange(stud,(Marks))

query3

arrange(stud,Name)

select(stud,Name)

/*show marks belongs to each class */

  filter (stud,Class=="fycs")%>%arrange(Marks)

filter (stud,Class=="Sycs")%>%arrange(Marks)

filter (stud,Class=="Tycs")%>%arrange(Marks)

mutate(stud,Perc=Marks/10)

/for max marks of each class/

  stud1<-group_by(stud,Class)

stud1

query2<-summarize(stud1,max(Marks))

query2
```

Excel File:



	A	B	C	D
1	Name	Class	Marks	
2	Gautami	Ty.Bsc.Cs	500	
3	Priya	Sy.Bsc.Cs	500	
4	Sharvi	Ty.Bsc.IT	500	
5				
6				

Output:



```
> /*show marks belongs to each class */
Error: unexpected '/' in "/"
>   filter (stud,Class=="fycs")%>%arrange(Marks)
[1] Name  Class Marks
<0 rows> (or 0-length row.names)
> filter (stud,Class=="Sycs")%>%arrange(Marks)
[1] Name  Class Marks
<0 rows> (or 0-length row.names)
> filter (stud,Class=="Tycs")%>%arrange(Marks)
[1] Name  Class Marks
<0 rows> (or 0-length row.names)
> mutate(stud,Perc=Marks/10)
     Name     Class Marks Perc
1 Gautami Ty.Bsc.Cs   500   50
2   Priya Sy.Bsc.Cs   500   50
3  Sharvi Ty.Bsc.IT   500   50
> /*for max marks of each class*/
Error: unexpected '/' in "/"
>   stud1<-group_by(stud,Class)
> stud1
# A tibble: 3 x 3
# Groups:   Class [3]
  Name    Class     Marks
  <chr>   <chr>     <int>
1 Gautami Ty.Bsc.Cs   500
2 Priya   Sy.Bsc.Cs   500
3 Sharvi  Ty.Bsc.IT   500
> query2<-summarize(stud1,max(Marks))
> query2
# A tibble: 3 x 2
  Class     `max(Marks)`
  <chr>            <int>
1 Sy.Bsc.Cs          500
2 Ty.Bsc.Cs          500
3 Ty.Bsc.IT          500
> |
```

Practical 2

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

Description:

MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (similar to JSON format).

SQL databases store data in tabular format. This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications. Modern applications are more networked, social and interactive than ever. Applications are storing more and more data and are accessing it at higher rates.

Relational Database Management System(RDBMS) **i**s not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable. If the database runs on a single server, then it will reach a scaling limit. NoSQL databases are more scalable and provide superior performance. MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

Features of MongoDB:

- **Document Oriented**: MongoDB stores the main subject in the minimal number of documents and not by breaking it up into multiple relational structures like RDBMS. For example, it stores all the information of a computer in a single document called Computer and not in distinct relational structures like CPU, RAM, Hard disk, etc.
- **Indexing**: Without indexing, a database would have to scan every document of a collection to select those that match the query which would be inefficient. So, for efficient searching Indexing is a must and MongoDB uses it to process huge volumes of data in very less time.
- **Scalability**: MongoDB scales horizontally using sharding (partitioning data across various servers). Data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that reside across many physical servers. Also, new machines can be added to a running database.
- **Replication and High Availability**: MongoDB increases the data availability with multiple copies of data on different servers. By providing redundancy, it protects the database from hardware failures. If one server goes down, the data can be retrieved easily from other active servers which also had the data stored on them.
- **Aggregation**: Aggregation operations process data records and return the computed results. It is similar to the GROUPBY clause in SQL. A few aggregation expressions are sum, avg, min, max, etc.

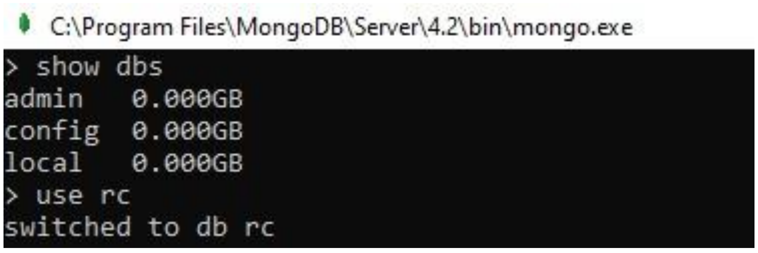
Where do we use MongoDB?

MongoDB is preferred over RDBMS in the following scenarios:

- **Big Data**: If you have huge amount of data to be stored in tables, think of MongoDB before RDBMS databases. MongoDB has built-in solution for partitioning and sharding your database.
- **Unstable Schema**: Adding a new column in RDBMS is hard whereas MongoDB is schema-less. Adding a new field does not effect old documents and will be very easy.
- **Distributed data** Since multiple copies of data are stored across different servers, recovery of data is instant and safe even if there is a hardware failure.

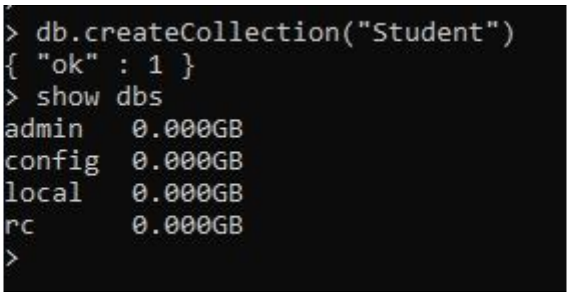
Output:

Show dbs



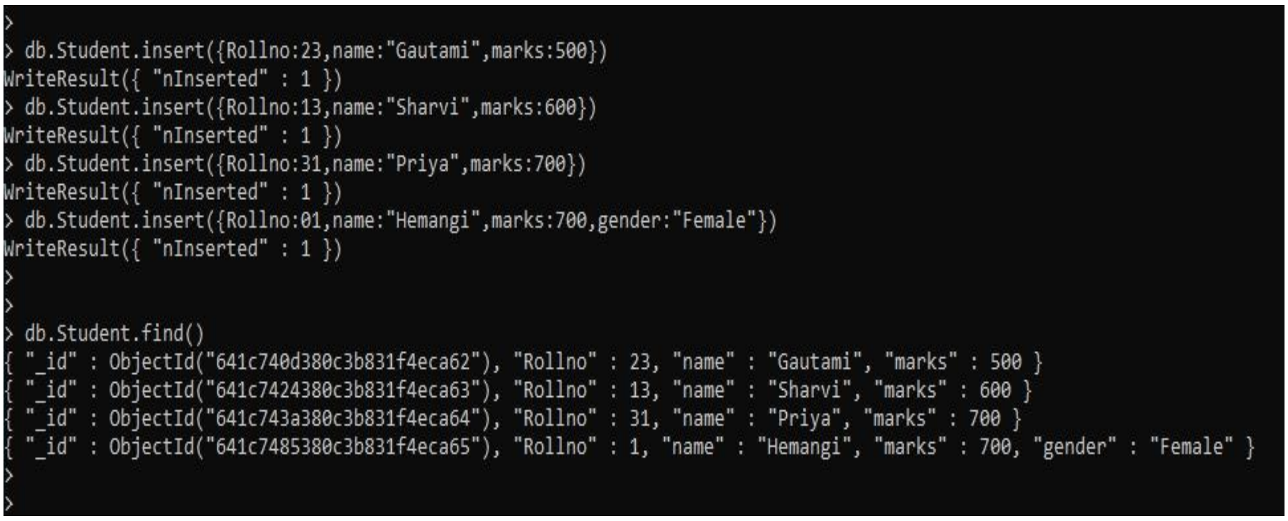
```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use rc
switched to db rc
```

To create database



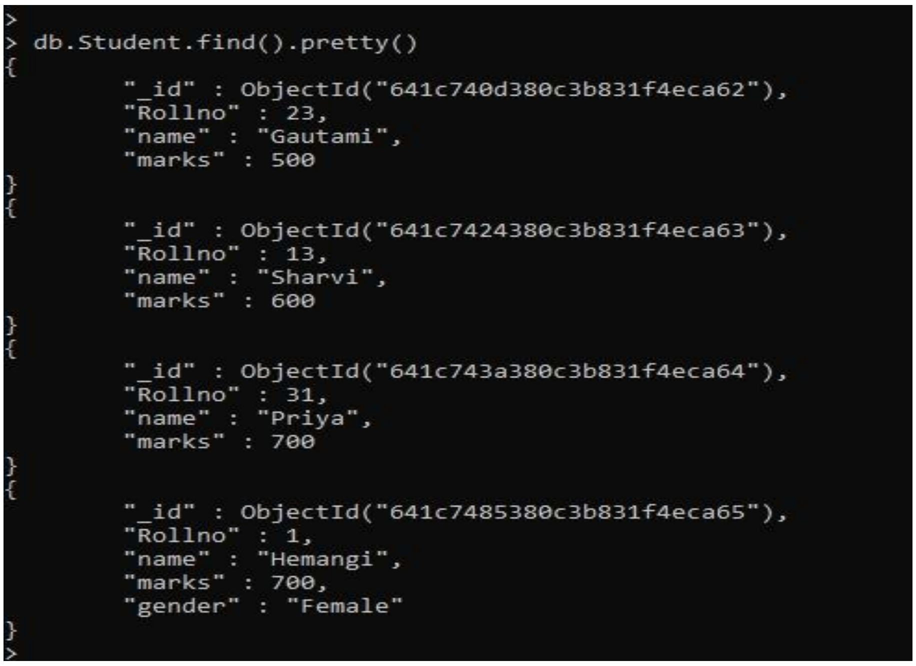
```
> db.createCollection("Student")
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
rc      0.000GB
>
```

Inserting Data in Database



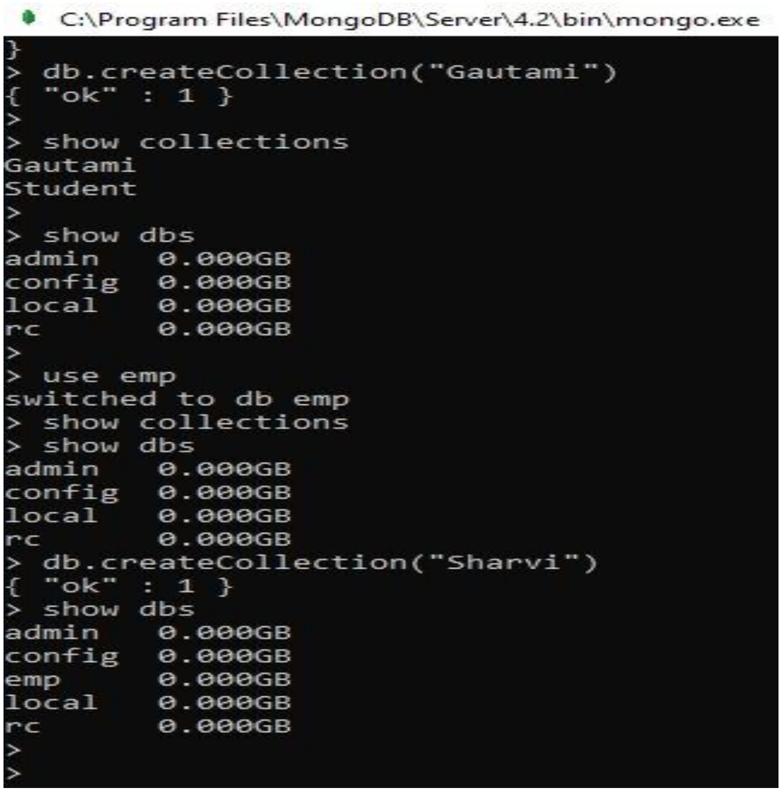
```
>
> db.Student.insert({Rollno:23,name:"Gautami",marks:500})
WriteResult({ "nInserted" : 1 })
> db.Student.insert({Rollno:13,name:"Sharvi",marks:600})
WriteResult({ "nInserted" : 1 })
> db.Student.insert({Rollno:31,name:"Priya",marks:700})
WriteResult({ "nInserted" : 1 })
> db.Student.insert({Rollno:01,name:"Hemangi",marks:700,gender:"Female"})
WriteResult({ "nInserted" : 1 })
>
>
> db.Student.find()
{ "_id" : ObjectId("641c740d380c3b831f4eca62"), "Rollno" : 23, "name" : "Gautami", "marks" : 500 }
{ "_id" : ObjectId("641c7424380c3b831f4eca63"), "Rollno" : 13, "name" : "Sharvi", "marks" : 600 }
{ "_id" : ObjectId("641c743a380c3b831f4eca64"), "Rollno" : 31, "name" : "Priya", "marks" : 700 }
{ "_id" : ObjectId("641c7485380c3b831f4eca65"), "Rollno" : 1, "name" : "Hemangi", "marks" : 700, "gender" : "Female" }
>
>
```

Listing

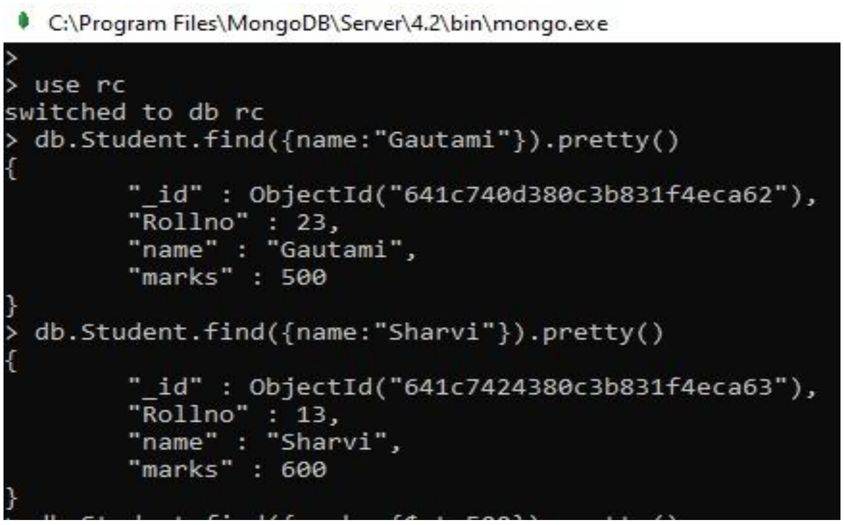


```
>
> db.Student.find().pretty()
{
        "_id" : ObjectId("641c740d380c3b831f4eca62"),
        "Rollno" : 23,
        "name" : "Gautami",
        "marks" : 500
}
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
{
        "_id" : ObjectId("641c743a380c3b831f4eca64"),
        "Rollno" : 31,
        "name" : "Priya",
        "marks" : 700
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
>
```

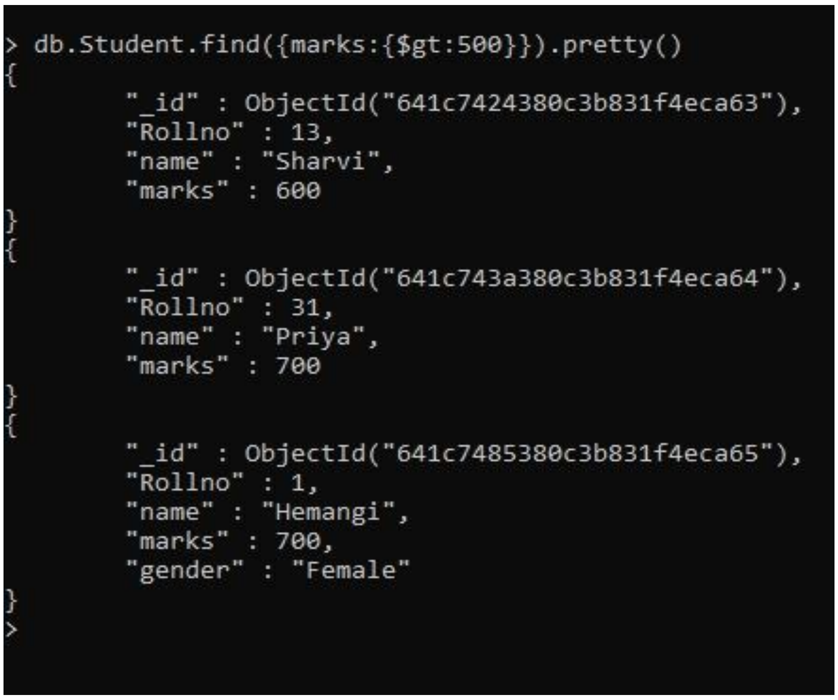
Creating Collection

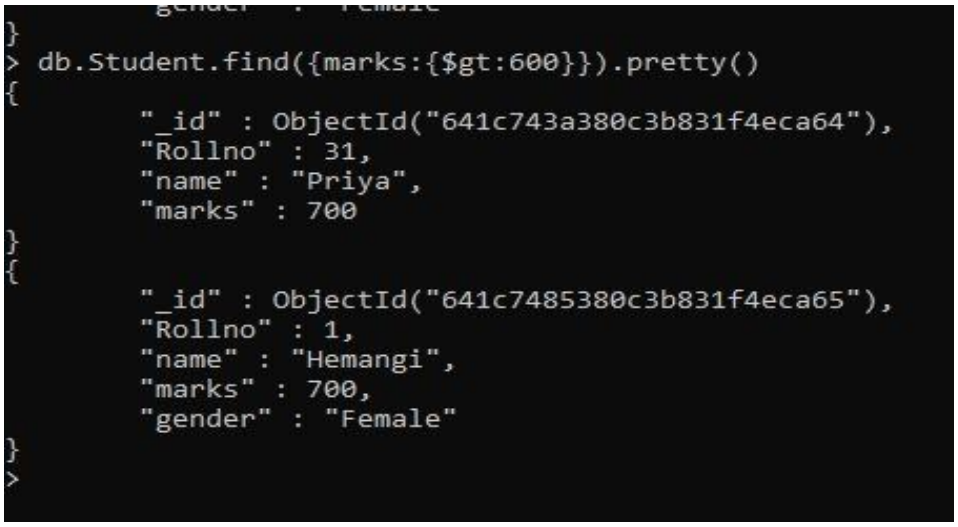


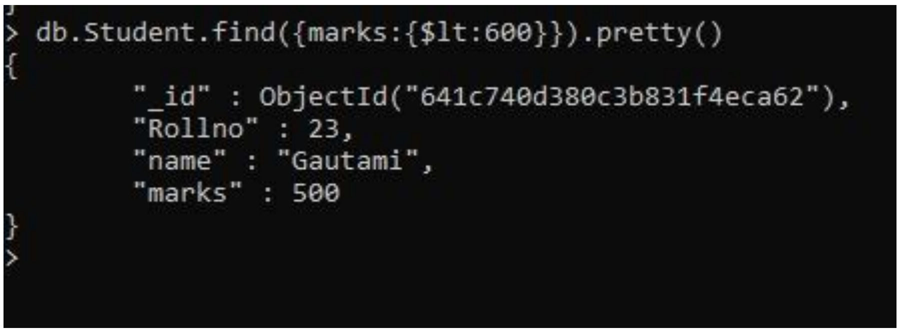
```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
}
> db.createCollection("Gautami")
{ "ok" : 1 }
>
> show collections
Gautami
Student
>
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
rc      0.000GB
>
> use emp
switched to db emp
> show collections
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
rc      0.000GB
> db.createCollection("Sharvi")
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
emp     0.000GB
local   0.000GB
rc      0.000GB
>
>
```


```
C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
>
> use rc
switched to db rc
> db.Student.find({name:"Gautami"}).pretty()
{
        "_id" : ObjectId("641c740d380c3b831f4eca62"),
        "Rollno" : 23,
        "name" : "Gautami",
        "marks" : 500
}
> db.Student.find({name:"Sharvi"}).pretty()
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
```

Using $gt & $lt

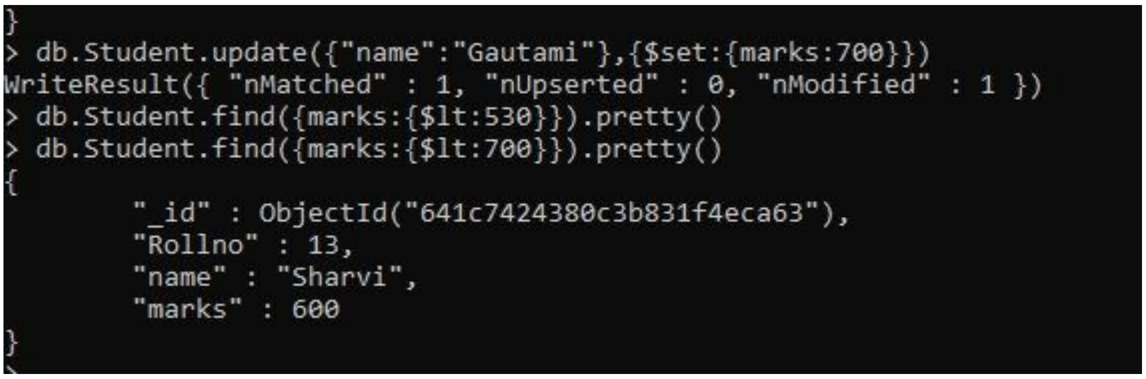

```
> db.Student.find({marks:{$gt:500}}).pretty()
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
{
        "_id" : ObjectId("641c743a380c3b831f4eca64"),
        "Rollno" : 31,
        "name" : "Priya",
        "marks" : 700
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
>
```


```
}
> db.Student.find({marks:{$gt:600}}).pretty()
{
        "_id" : ObjectId("641c743a380c3b831f4eca64"),
        "Rollno" : 31,
        "name" : "Priya",
        "marks" : 700
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
>
```

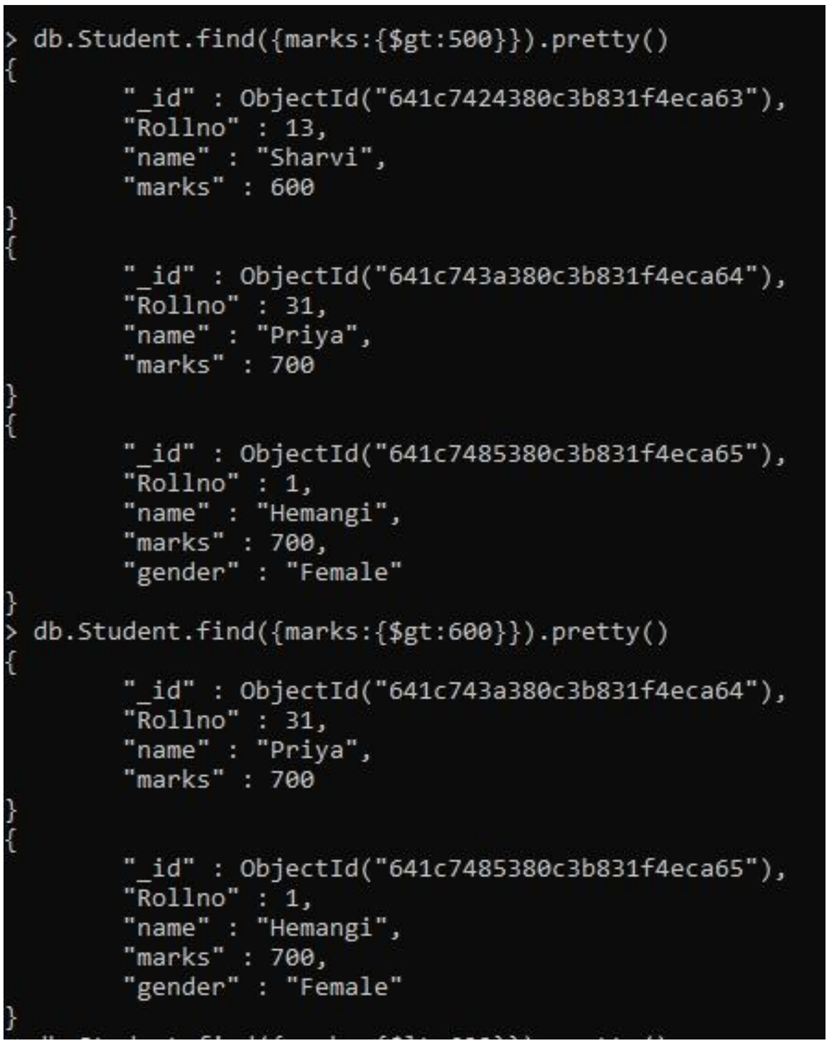


```
> db.Student.find({marks:{$lt:600}}).pretty()
{
        "_id" : ObjectId("641c740d380c3b831f4eca62"),
        "Rollno" : 23,
        "name" : "Gautami",
        "marks" : 500
}
>
```

Updating the Data

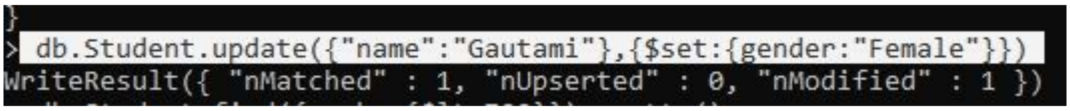


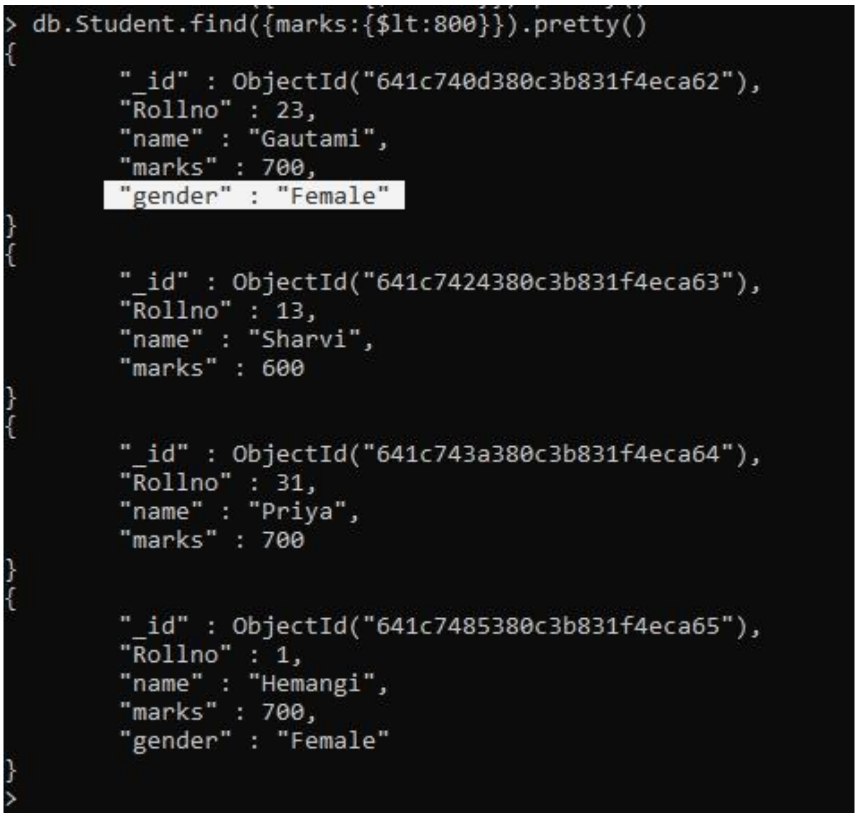
```
}
> db.Student.update({"name":"Gautami"},{$set:{marks:700}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({marks:{$lt:530}}).pretty()
> db.Student.find({marks:{$lt:700}}).pretty()
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
```



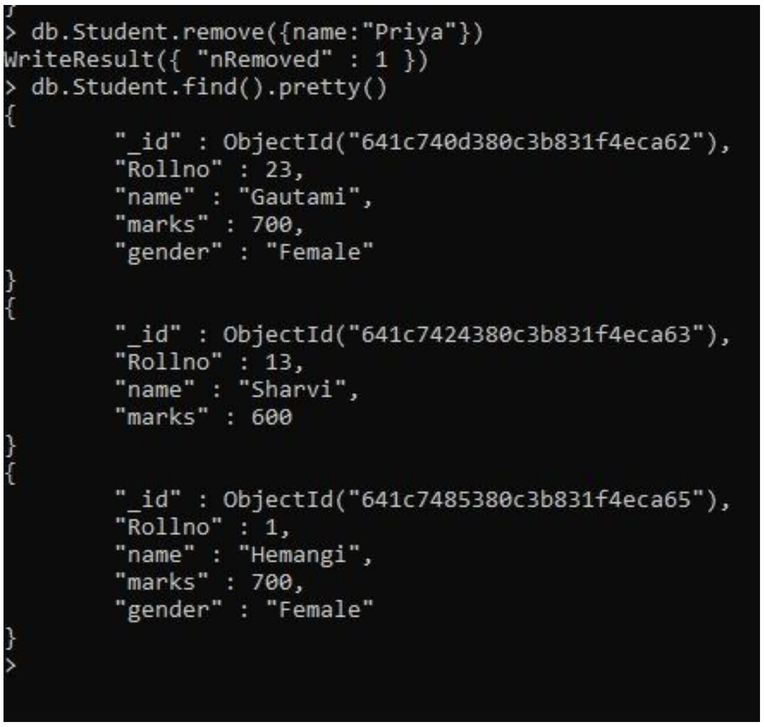
```
> db.Student.find({marks:{$gt:500}}).pretty()
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
{
        "_id" : ObjectId("641c743a380c3b831f4eca64"),
        "Rollno" : 31,
        "name" : "Priya",
        "marks" : 700
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
> db.Student.find({marks:{$gt:600}}).pretty()
{
        "_id" : ObjectId("641c743a380c3b831f4eca64"),
        "Rollno" : 31,
        "name" : "Priya",
        "marks" : 700
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
```

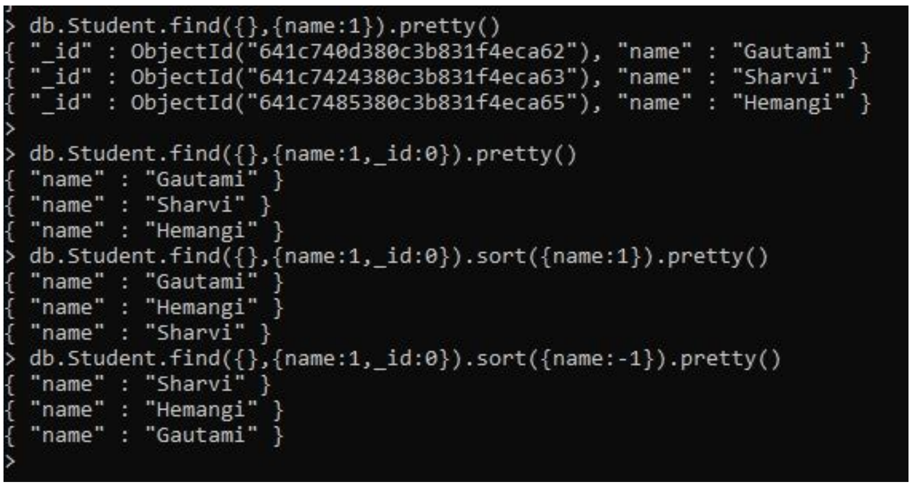
Updating the Data


```
}
> db.Student.update({"name":"Gautami"},{$set:{gender:"Female"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```


```
> db.Student.find({marks:{$lt:800}}).pretty()
{
        "_id" : ObjectId("641c740d380c3b831f4eca62"),
        "Rollno" : 23,
        "name" : "Gautami",
        "marks" : 700,
        "gender" : "Female"
}
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
{
        "_id" : ObjectId("641c743a380c3b831f4eca64"),
        "Rollno" : 31,
        "name" : "Priya",
        "marks" : 700
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
>
```

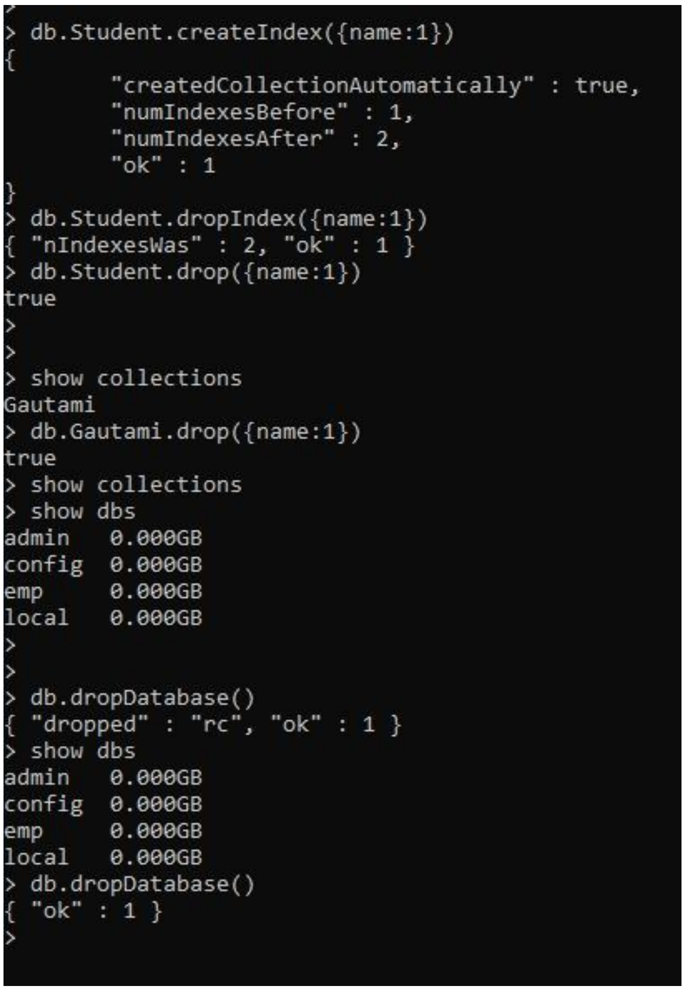
Removing Data


```
> db.Student.remove({name:"Priya"})
WriteResult({ "nRemoved" : 1 })
> db.Student.find().pretty()
{
        "_id" : ObjectId("641c740d380c3b831f4eca62"),
        "Rollno" : 23,
        "name" : "Gautami",
        "marks" : 700,
        "gender" : "Female"
}
{
        "_id" : ObjectId("641c7424380c3b831f4eca63"),
        "Rollno" : 13,
        "name" : "Sharvi",
        "marks" : 600
}
{
        "_id" : ObjectId("641c7485380c3b831f4eca65"),
        "Rollno" : 1,
        "name" : "Hemangi",
        "marks" : 700,
        "gender" : "Female"
}
>
```



```
> db.Student.find({},{name:1}).pretty()
{ "_id" : ObjectId("641c740d380c3b831f4eca62"), "name" : "Gautami" }
{ "_id" : ObjectId("641c7424380c3b831f4eca63"), "name" : "Sharvi" }
{ "_id" : ObjectId("641c7485380c3b831f4eca65"), "name" : "Hemangi" }
>
> db.Student.find({},{name:1,_id:0}).pretty()
{ "name" : "Gautami" }
{ "name" : "Sharvi" }
{ "name" : "Hemangi" }
> db.Student.find({},{name:1,_id:0}).sort({name:1}).pretty()
{ "name" : "Gautami" }
{ "name" : "Hemangi" }
{ "name" : "Sharvi" }
> db.Student.find({},{name:1,_id:0}).sort({name:-1}).pretty()
{ "name" : "Sharvi" }
{ "name" : "Hemangi" }
{ "name" : "Gautami" }
>
```

<u>Creating and Dropping the Index / Droping the Database</u>



```
> db.Student.createIndex({name:1})
{
        "createdCollectionAutomatically" : true,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.Student.dropIndex({name:1})
{ "nIndexesWas" : 2, "ok" : 1 }
> db.Student.drop({name:1})
true
>
>
> show collections
Gautami
> db.Gautami.drop({name:1})
true
> show collections
> show dbs
admin   0.000GB
config  0.000GB
emp     0.000GB
local   0.000GB
>
>
> db.dropDatabase()
{ "dropped" : "rc", "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
emp     0.000GB
local   0.000GB
> db.dropDatabase()
{ "ok" : 1 }
>
```

Practical 3

Aim: Practical of Principal Component Analysis

Description:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

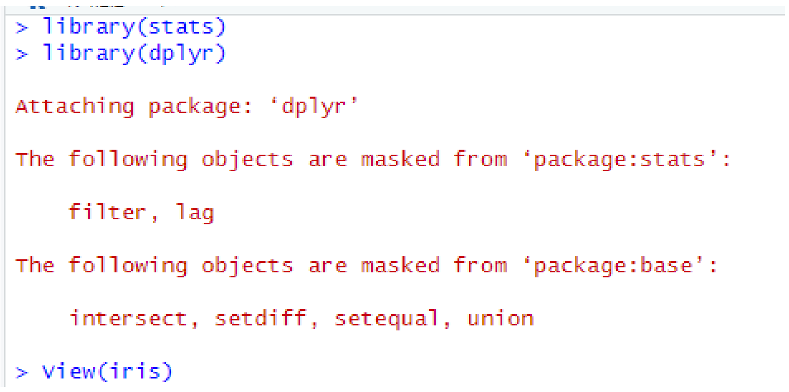
Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

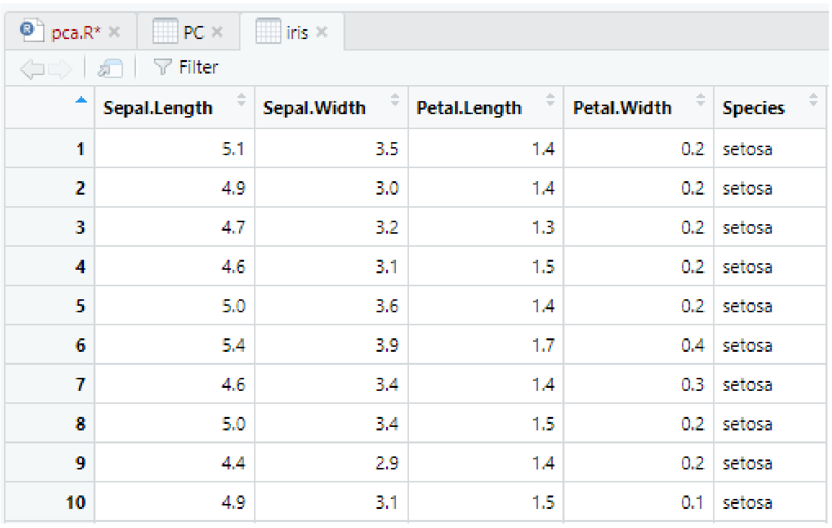
Code:

```
#install packages

install.packages("stats")

install.packages("dplyr")

#load required libraries

library(stats)

library(dplyr)

#iris dataset

View(iris)

#Unsupervised learning

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

#Check PCA eligibility

cor(mydata)

mean(cor(mydata))

#Principal component analysis

PCA = princomp(mydata)

#PC loading

PCA$loadings

#Principal Components

PC=PCA$scores

View(PC)

cor(PC)
```

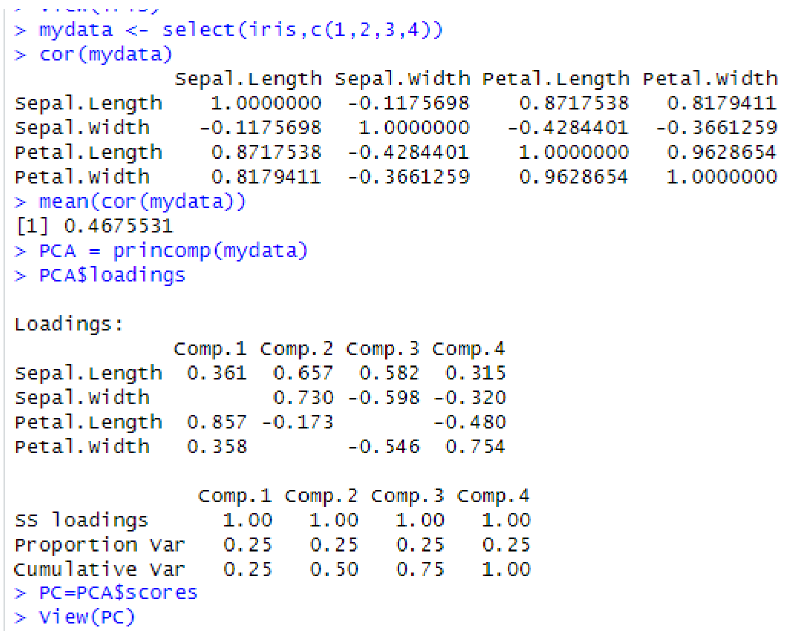
Output:



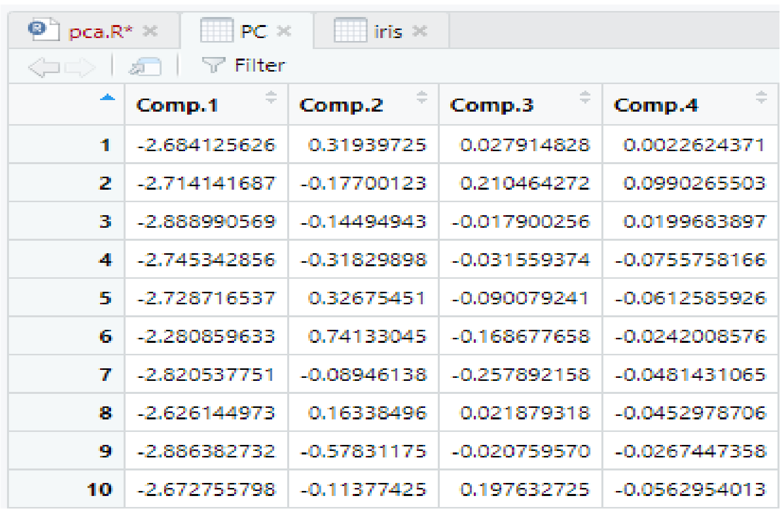
```
> library(stats)
> library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

> View(iris)
```



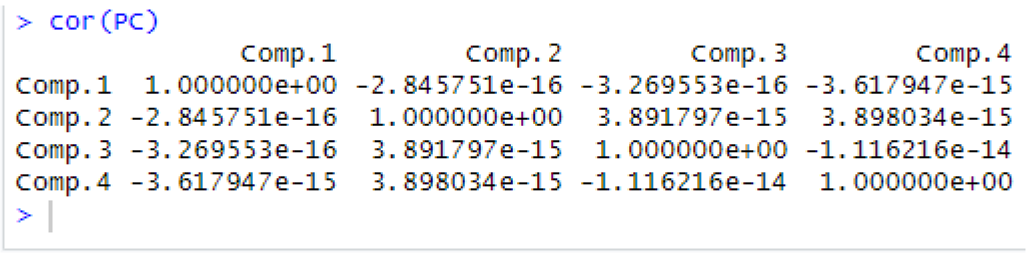
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1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



```
> mydata <- select(iris,c(1,2,3,4))
> cor(mydata)
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
> mean(cor(mydata))
[1] 0.4675531
> PCA = princomp(mydata)
> PCA$loadings

Loadings:
             Comp.1 Comp.2 Comp.3 Comp.4
Sepal.Length  0.361  0.657  0.582  0.315
Sepal.Width          0.730 -0.598 -0.320
Petal.Length  0.857 -0.173        -0.480
Petal.Width   0.358        -0.546  0.754

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
> PC=PCA$scores
> View(PC)
```



	Comp.1	Comp.2	Comp.3	Comp.4
1	-2.684125626	0.31939725	0.027914828	0.0022624371
2	-2.714141687	-0.17700123	0.210464272	0.0990265503
3	-2.888990569	-0.14494943	-0.017900256	0.0199683897
4	-2.745342856	-0.31829898	-0.031559374	-0.0755758166
5	-2.728716537	0.32675451	-0.090079241	-0.0612585926
6	-2.280859633	0.74133045	-0.168677658	-0.0242008576
7	-2.820537751	-0.08946138	-0.257892158	-0.0481431065
8	-2.626144973	0.16338496	0.021879318	-0.0452978706
9	-2.886382732	-0.57831175	-0.020759570	-0.0267447358
10	-2.672755798	-0.11377425	0.197632725	-0.0562954013



```
> cor(PC)
              Comp.1        Comp.2        Comp.3        Comp.4
Comp.1  1.000000e+00 -2.845751e-16 -3.269553e-16 -3.617947e-15
Comp.2 -2.845751e-16  1.000000e+00  3.891797e-15  3.898034e-15
Comp.3 -3.269553e-16  3.891797e-15  1.000000e+00 -1.116216e-14
Comp.4 -3.617947e-15  3.898034e-15 -1.116216e-14  1.000000e+00
> 
```

Practical no: 4

Aim: Practical of Clustering

Description

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behaviour, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML systems can use this id to simplify the processing of large and complex datasets.

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

The clustering technique is commonly used for statistical data analysis.

clustering technique with the real-world example of Mall: When we visit any shopping mall, we can observe that the things with similar usage are grouped together. Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., are grouped in separate sections, so that we can easily find out the things. The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

○ Market Segmentation

○ Statistical data analysis

○ Social network analysis

○ Image segmentation

○ Anomaly detection, etc.

Apart from these general usages, it is used by Amazon in its recommendation system to provide the recommendations as per the past search of products.

Netflix also uses this technique to recommend the movies and web-series to its users as per the watch history.

The clustering methods are broadly divided into Hard clustering (data point belongs to only one group) and Soft Clustering (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning?

1. Partitioning Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the centroid-based method. The most common example of partitioning clustering is the K-Means Clustering algorithm.

In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.

2. Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.

3. Distribution Model-Based Clustering

In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly Gaussian Distribution.

The example of this type is the Expectation-Maximization Clustering algorithm that uses Gaussian Mixture Models (GMM).

4. Hierarchical Clustering

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a dendrogram. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the Agglomerative Hierarchical algorithm.

5. Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster.

Fuzzy C-means algorithm is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

K-Means algorithm: The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of O(n).

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabelled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabelled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

<u>The k-means clustering algorithm mainly performs two tasks:</u>

○ determines the best value for K centred points or centroids by an iterative process.

○ assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has data points with some commonalities, and it is away from other clusters.

<u>The working of the K-Means algorithm is explained in the below steps:</u>

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be different from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which mean assign each data point to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH. Step-7: The model is ready.

Applications of clustering

Below are some commonly known applications of clustering technique in Machine Learning:

○ **In Identification of Cancer Cells**: The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.

○ **In Search Engines**: Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurate result of a query depends on the quality of the clustering algorithm used.

○ **Customer Segmentation**: It is used in market research to segment the customers based on their choice and preferences.

○ **In Biology**: It is used in the biology stream to classify different species of plants and animals using the image recognition technique.

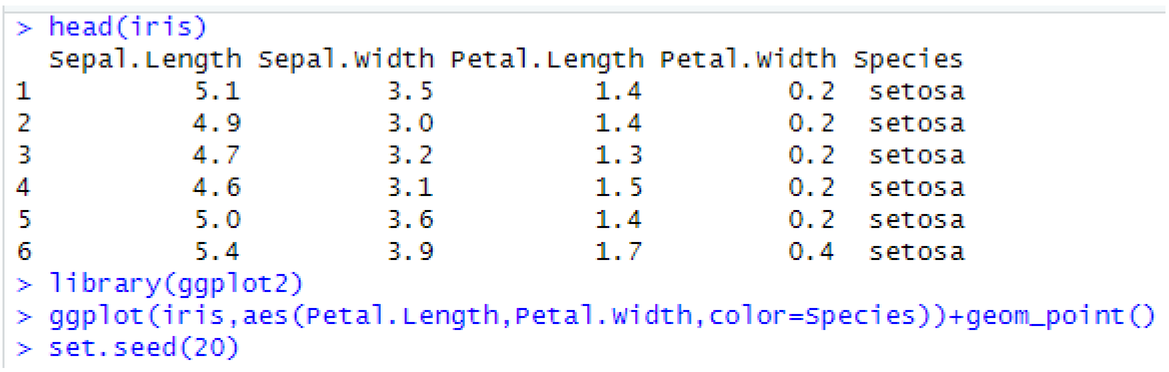
○ **In Land Use**: The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.

Code:

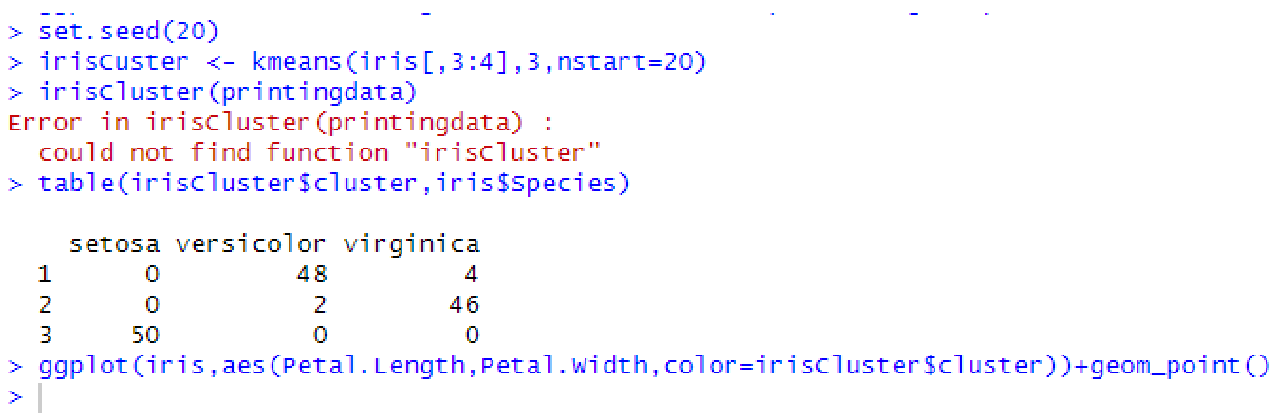
a.)

```
head(iris)

library(ggplot2)

ggplot(iris,aes(Petal.Length,Petal.Width,color=species))+geom_point()

set.seed(20)

irisCluster <- kmeans(iris[,3:4],3,nstart=20)

irisCluster(printingdata)

table(irisCluster$cluster,iris$Species)

ggplot(iris,aes(Petal.Length,Petal.Width,color=irisCluster$cluster))+geom_point()
```

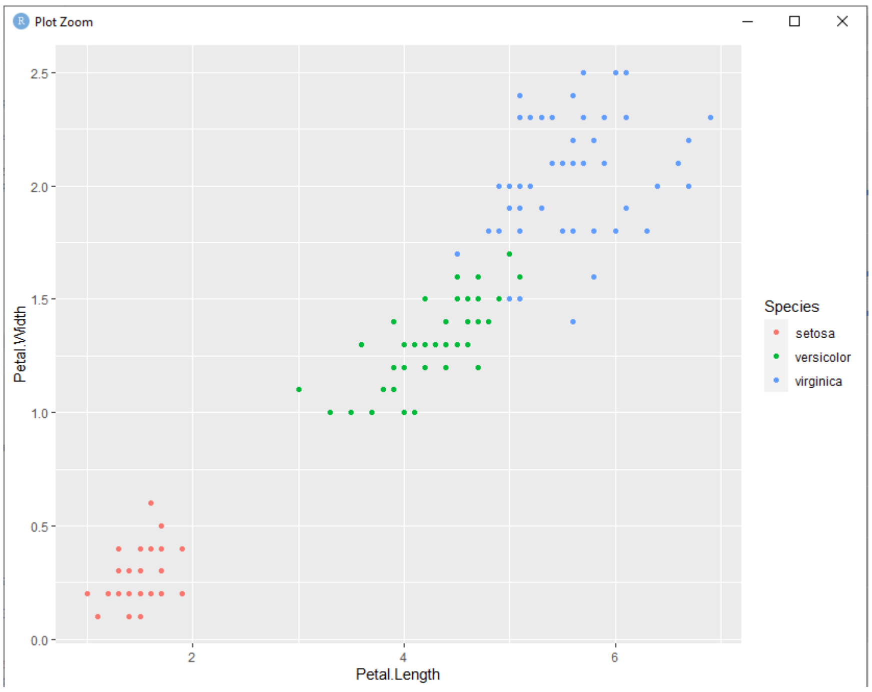
Output:

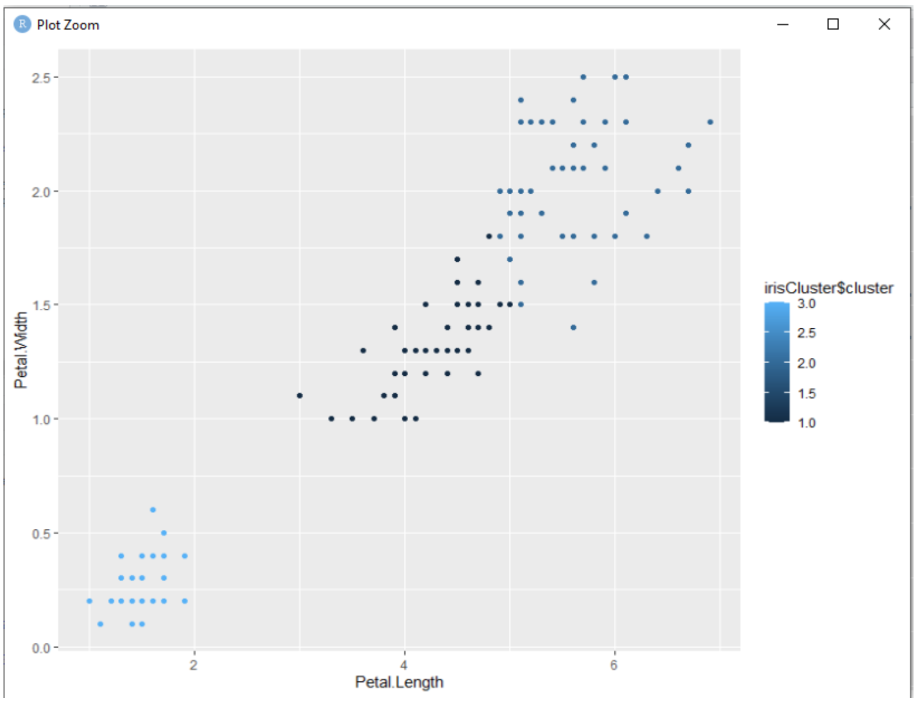


```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> library(ggplot2)
> ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point()
> set.seed(20)
```



```
> set.seed(20)
> irisCuster <- kmeans(iris[,3:4],3,nstart=20)
> irisCluster(printingdata)
Error in irisCluster(printingdata) : 
  could not find function "irisCluster"
> table(irisCluster$cluster,iris$Species)
   
    setosa versicolor virginica
  1      0         48         4
  2      0          2        46
  3     50          0         0
> ggplot(iris,aes(Petal.Length,Petal.Width,color=irisCluster$cluster))+geom_point()
> 
```





b.)

head(iris)

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

plot(clusters)

clusterCut<-cutree(cluster,3)

table(clusterCut,iris$Species)

cluster<-hclust(dist(iris[,3:4]),method='average')

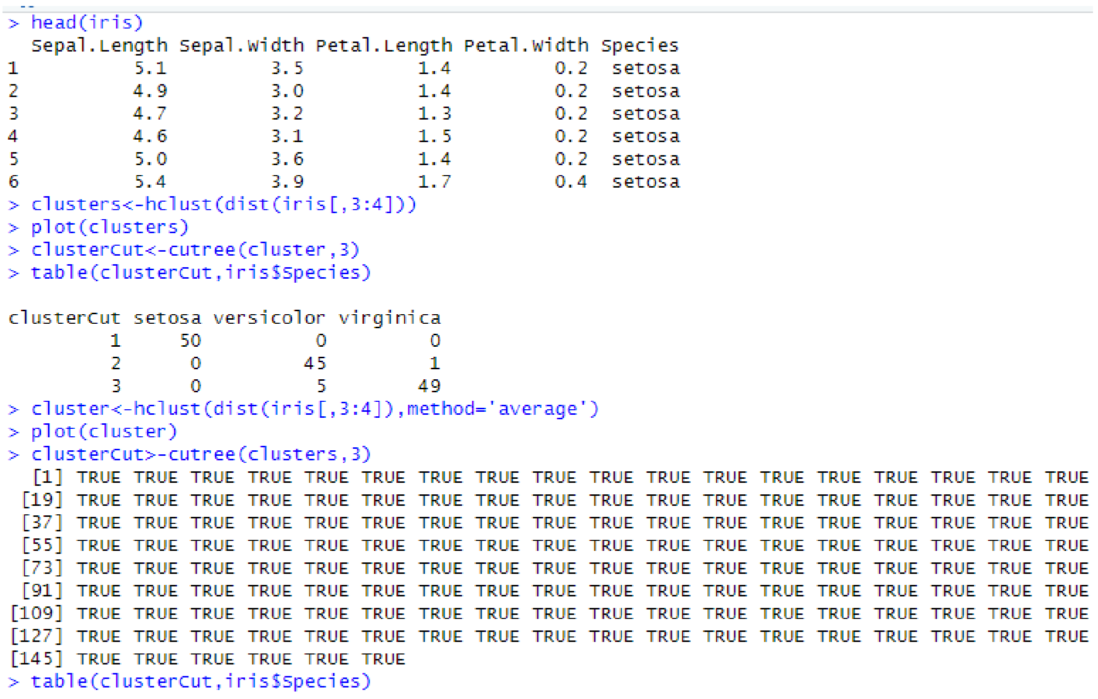
plot(cluster)

clusterCut>-cutree(clusters,3)

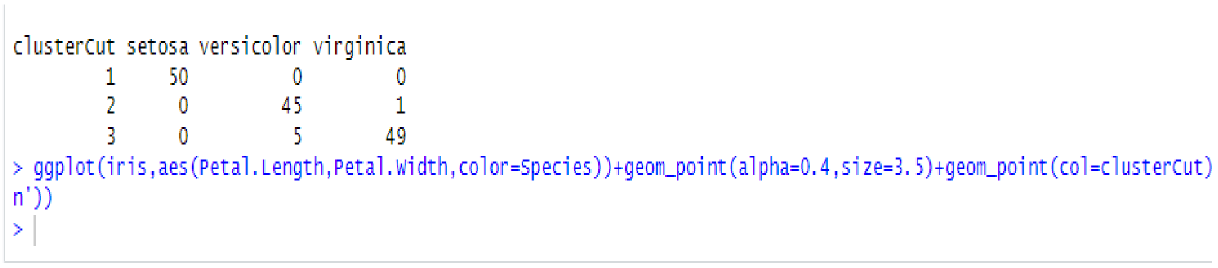
table(clusterCut,iris$Species)

ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point(alpha=0.4,size=3.5)+geom_point(
col=clusterCut)+scale_color_manual(values=c('black','red','green'))

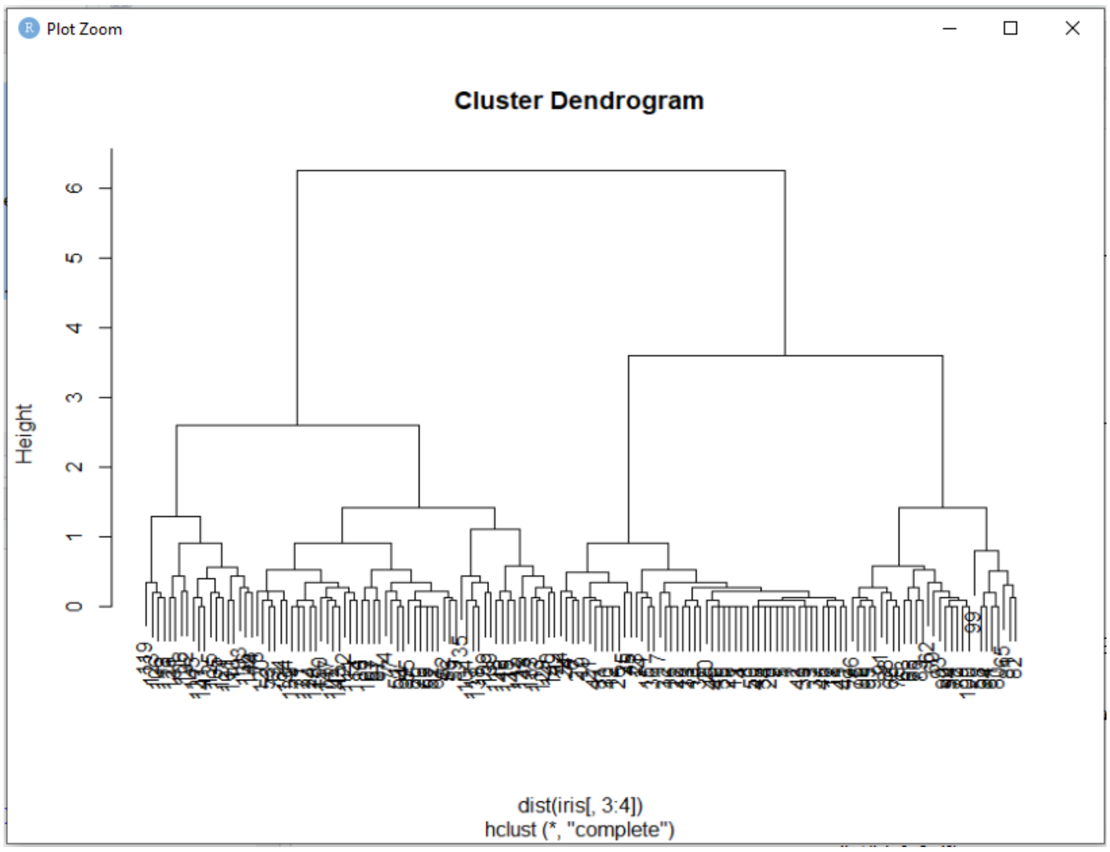
Output:

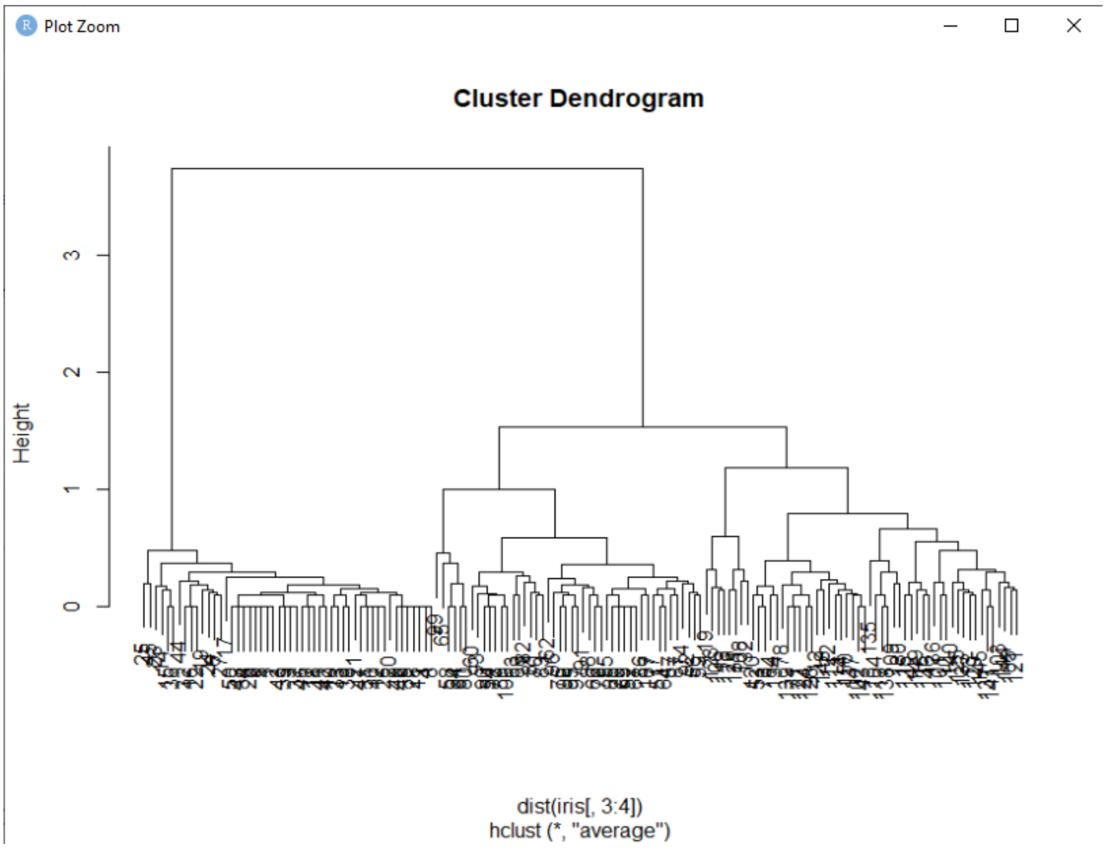


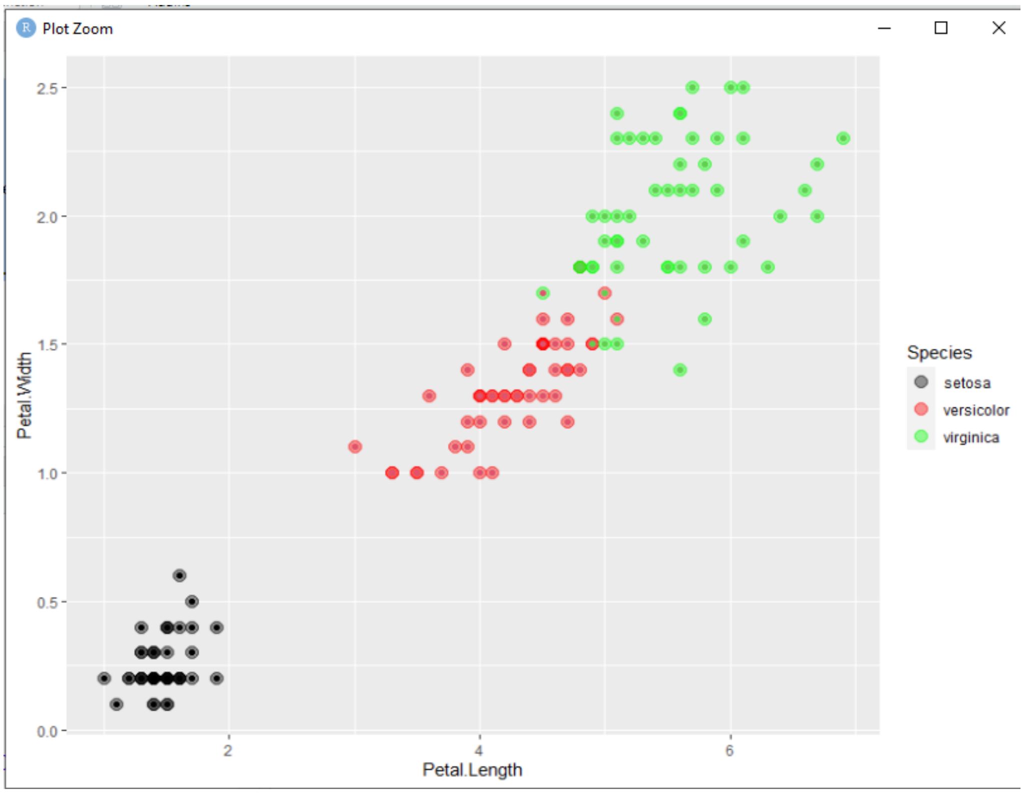
```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> clusters<-hclust(dist(iris[,3:4]))
> plot(clusters)
> clusterCut<-cutree(cluster,3)
> table(clusterCut,iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         45         1
         3      0          5        49
> cluster<-hclust(dist(iris[,3:4]),method='average')
> plot(cluster)
> clusterCut>-cutree(clusters,3)
  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [19] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [37] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [55] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [73] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [91] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[109] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[127] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[145] TRUE TRUE TRUE TRUE TRUE TRUE
> table(clusterCut,iris$Species)
```



```
clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         45         1
         3      0          5        49
> ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point(alpha=0.4,size=3.5)+geom_point(col=clusterCut)
n'))
> 
```




Plot Zoom
Cluster Dendrogram
Height
0
1
2
3
dist(iris[, 3:4])
hclust (*, "average")


Plot Zoom
Petal.Width
Petal.Length
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6
Species
setosa
versicolor
virginica

c.)

head(iris)

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

plot(clusters)

clusterCut<-cutree(cluster,3)

table(clusterCut,iris$Species)

cluster<-hclust(dist(iris[,3:4]),method='single')

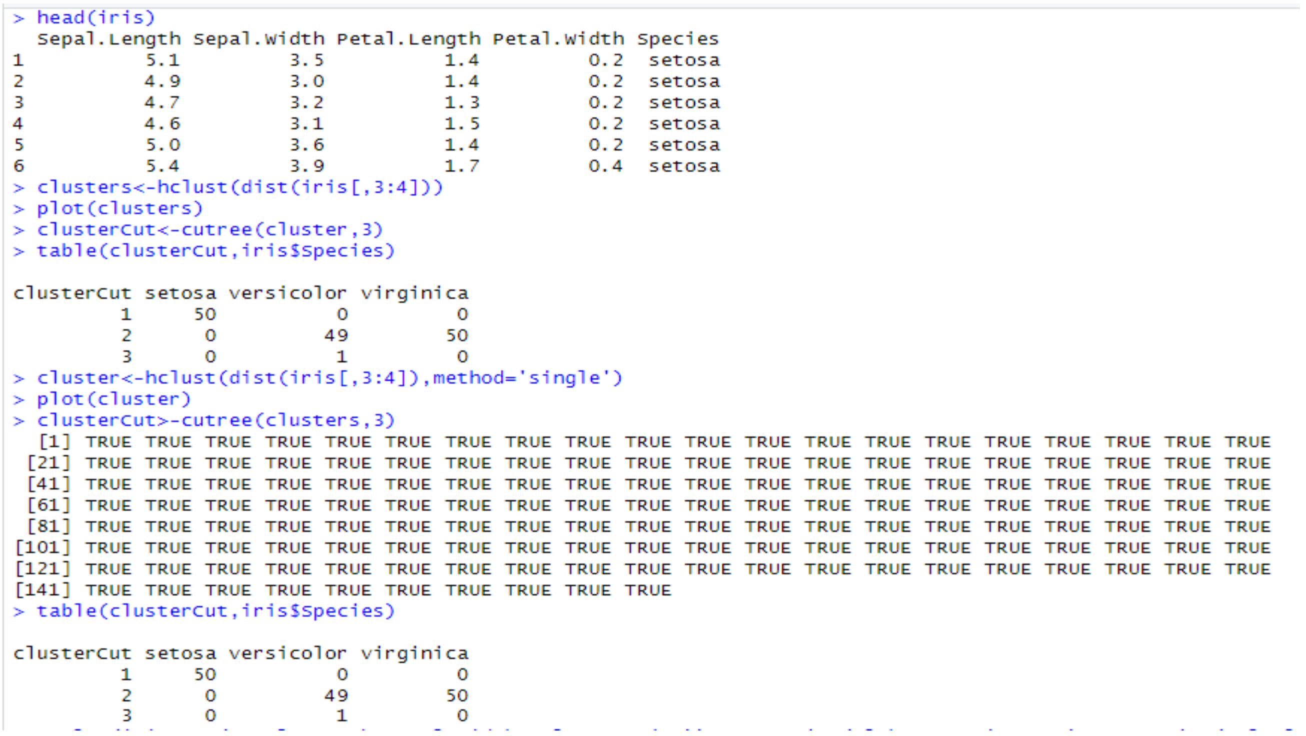
plot(cluster)

clusterCut>-cutree(clusters,3)

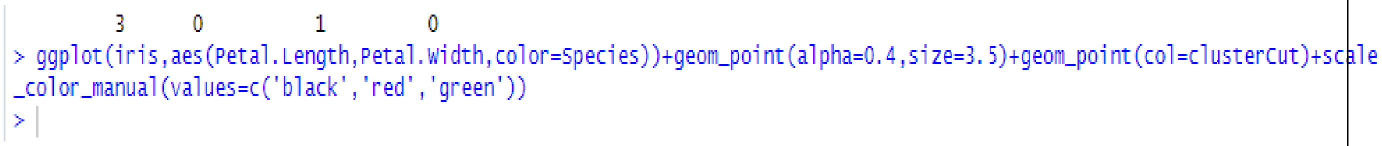
table(clusterCut,iris$Species)

ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point(alpha=0.4,size=3.5)+geom_point(col=clusterCut)+scale_color_manual(values=c('black','red','green'))

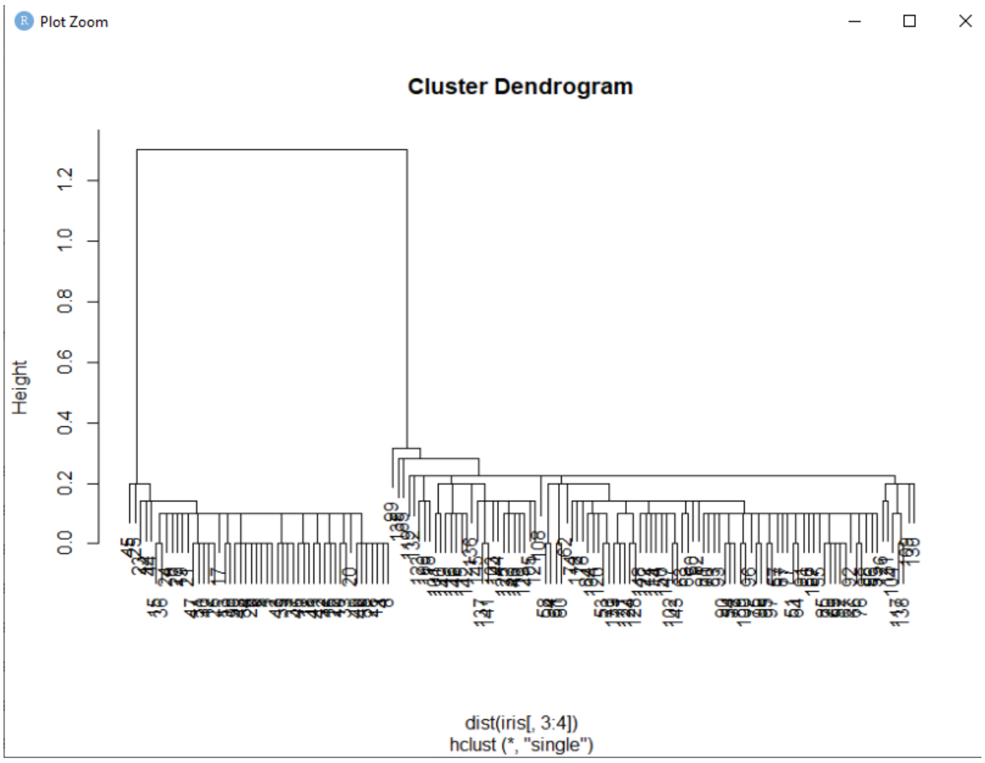
Output:

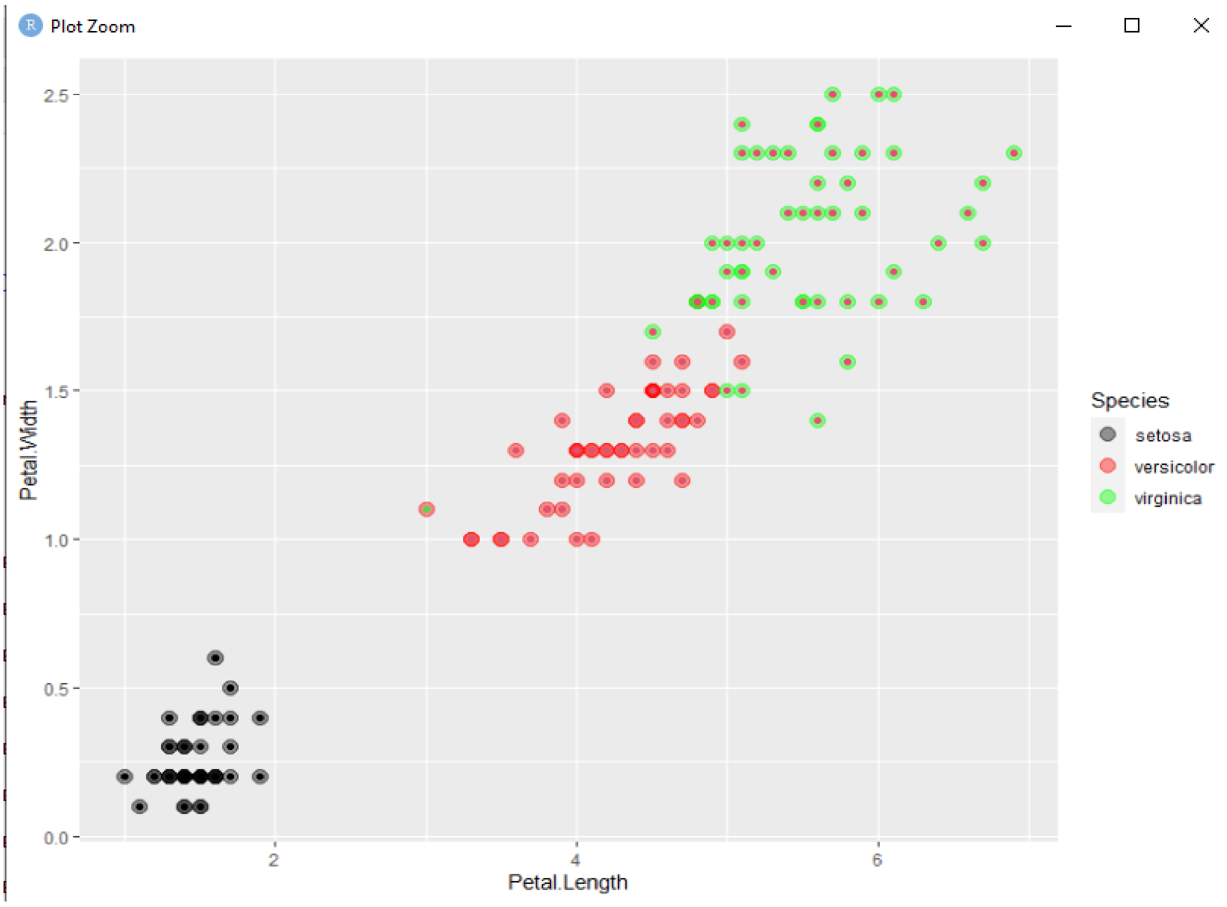


```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> clusters<-hclust(dist(iris[,3:4]))
> plot(clusters)
> clusterCut<-cutree(cluster,3)
> table(clusterCut,iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         49        50
         3      0          1         0
> cluster<-hclust(dist(iris[,3:4]),method='single')
> plot(cluster)
> clusterCut>-cutree(clusters,3)
  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [21] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [41] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
 [81] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[101] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[121] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[141] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> table(clusterCut,iris$Species)

clusterCut setosa versicolor virginica
         1     50          0         0
         2      0         49        50
         3      0          1         0
```



```
         3      0          1         0
> ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point(alpha=0.4,size=3.5)+geom_point(col=clusterCut)+scale
_color_manual(values=c('black','red','green'))
> 
```

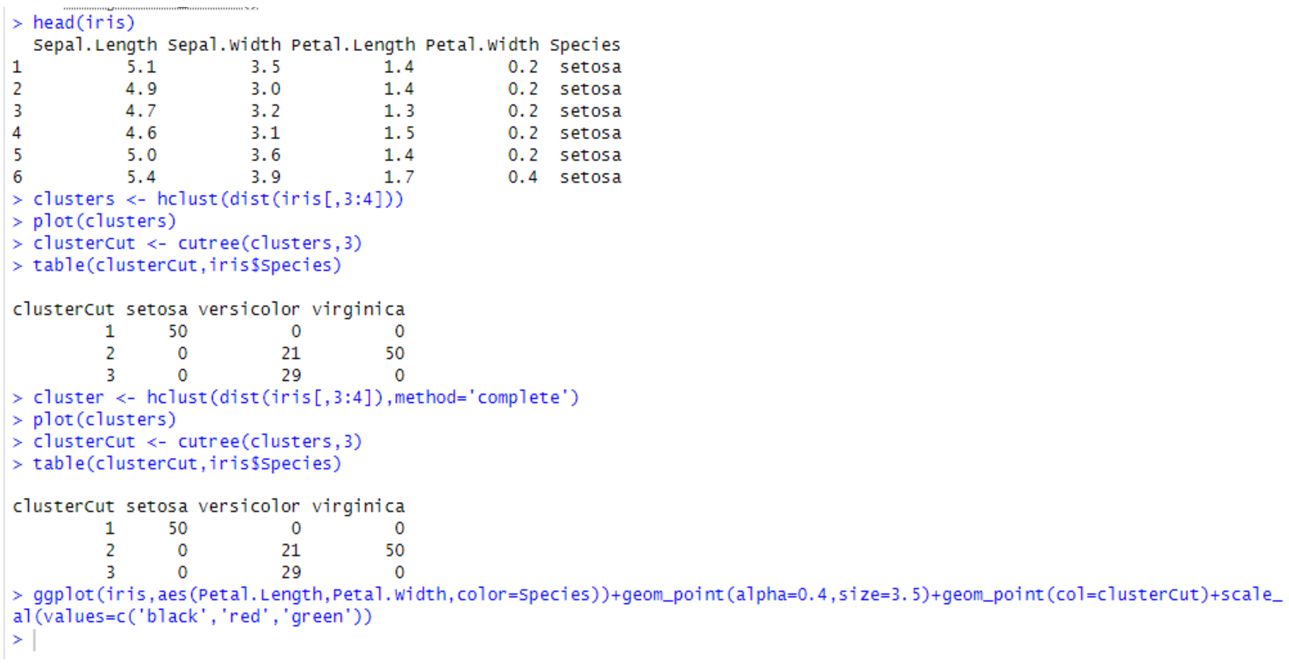

Plot Zoom
Cluster Dendrogram
Height
dist(iris[, 3:4])
hclust (*, "single")


Plot Zoom
Petal.Width
Petal.Length
Species
setosa
versicolor
virginica

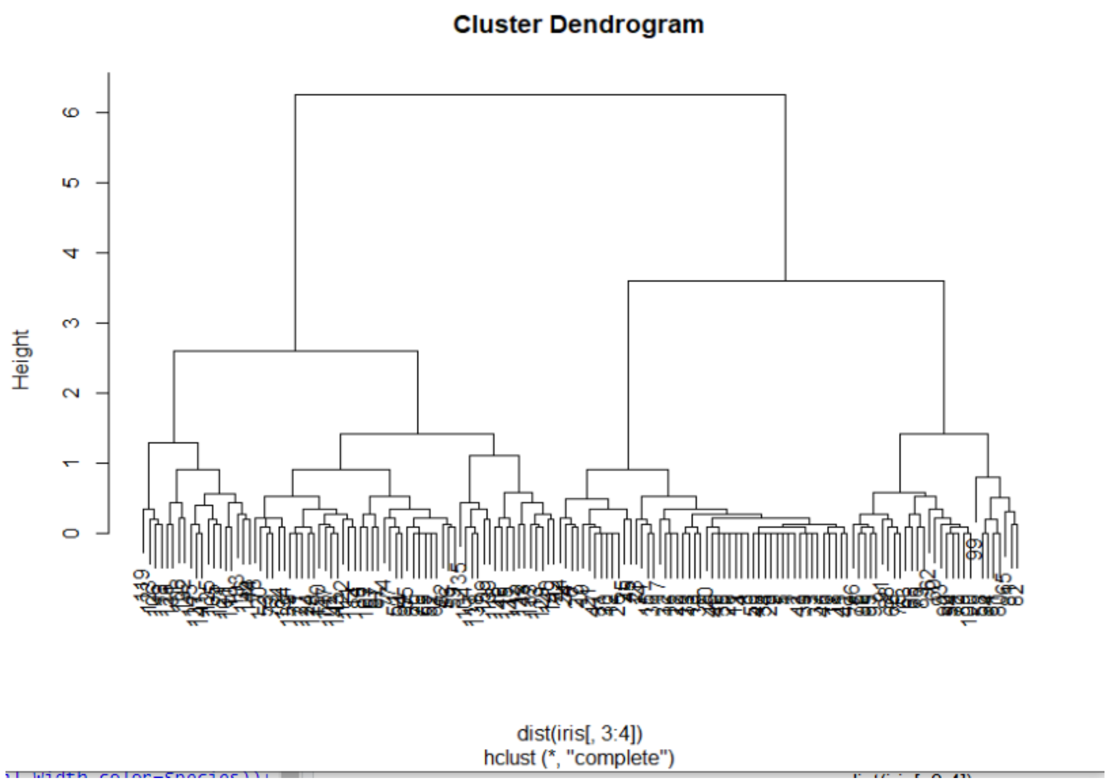
d.)

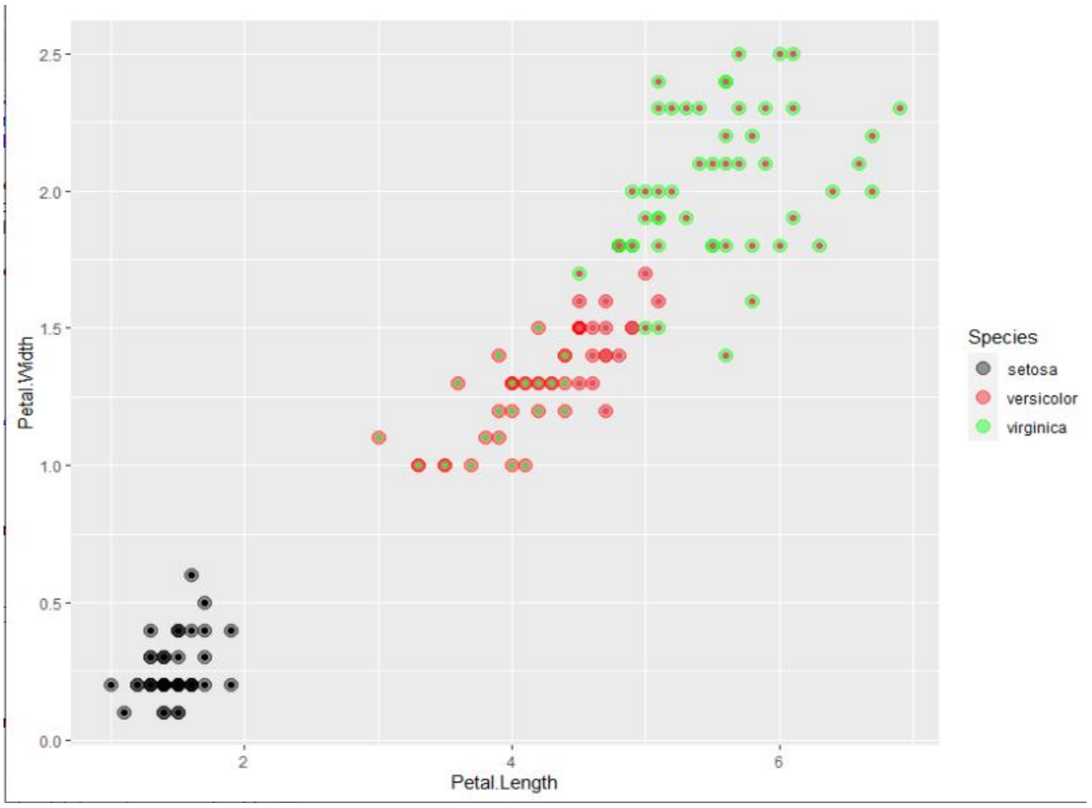
```
 head(iris)

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

plot(clusters)

clusterCut <- cutree(clusters,3)

table(clusterCut,iris$Species)

cluster <- hclust(dist(iris[,3:4]),method='complete')

plot(clusters)

clusterCut <- cutree(clusters,3)

table(clusterCut,iris$Species)

ggplot(iris,aes(Petal.Length,Petal.Width,color=iris$Species))+geom_point(alpha=0.4,size=3.5)=geom_p
oint(col=clusterCut)+scale_color_manual(values = c('black','red','green')
```

Output:



```
> 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
> 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
> cluster <- hclust(dist(iris[,3:4]),method='complete')
> 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
> ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point(alpha=0.4,size=3.5)+geom_point(col=clusterCut)+scale_
al(values=c('black','red','green'))
> 
```


Cluster Dendrogram
Height
0
1
2
3
4
5
6
dist(iris[, 3:4])
hclust (*, "complete")


Petal.Width
0.0
0.5
1.0
1.5
2.0
2.5
Petal.Length
2
4
6
Species
setosa
versicolor
virginica

Practical 5

Aim: Practical of Time-series forecasting

Description

Any metric which is measured over regular time intervals creates a time series. Analysis of time series is commercially important due to industrial necessity and relevance, especially with respect to the forecasting (demand, supply, and sale, etc.). A series of data points in which each data point is associated with a timestamp is knownas time series.

The stock price at different points in a day in the stock market is the simplest example of the time series. The amount of rainfall in an area in different months of the year is another example of it. R provides several functions for creating, manipulating, and plotting time series data. In the R-object, the time series data is known as the time-series object. It is just like a vector or data frame.

R provides ts() function for creating a Time Series.There is the following syntax of the ts() function:

```
Timeseries_object_name <- ts(data, start, end, frequency)
```

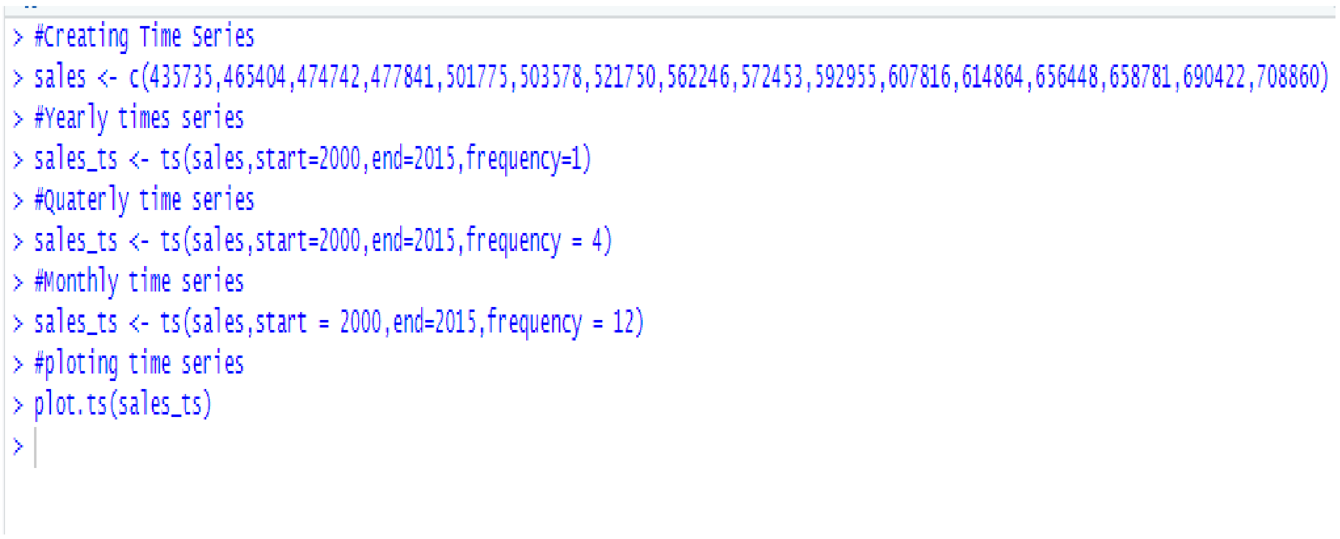
Sr No	**Parameter**	**Description**
1.	**data**	**It is a vector or matrix which contains the value used in time series.**
2.	**start**	**It is the start time for the first observation**
3.	**end**	**It is the end time for the last observation**
4.	**frequency**	**It specifies the number of observations per unit time.**

Code:

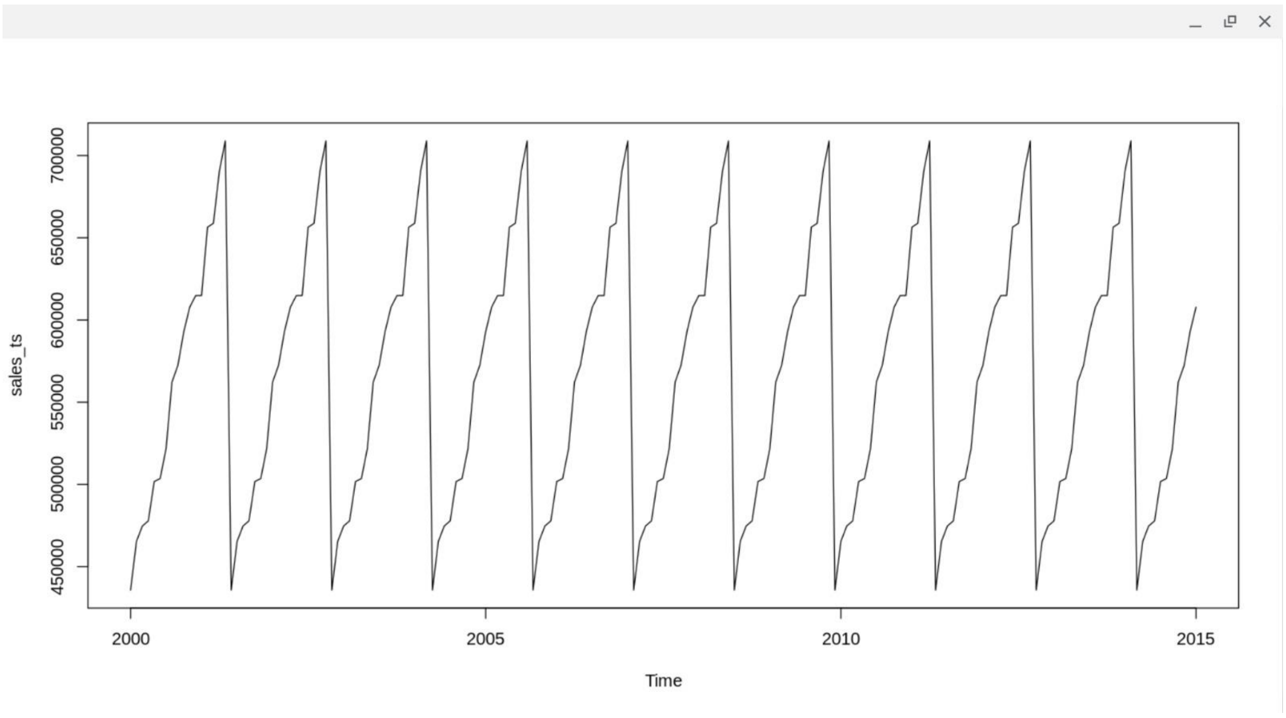
a.)

```
#Creating Time Series

sales                                                                                                    <-
c(435735,465404,474742,477841,501775,503578,521750,562246,572453,592955,607816,6148
64,656448,658781,690422,708860)

#Yearly times series

sales_ts <- ts(sales,start=2000,end=2015,frequency=1)

#Quaterly time series

sales_ts <- ts(sales,start=2000,end=2015,frequency = 4)

#Monthly time series

sales_ts <- ts(sales,start = 2000,end=2015,frequency = 12)

#ploting time series

plot.ts(sales_ts)
```

Output:



```
> #Creating Time Series
> sales <- c(435735,465404,474742,477841,501775,503578,521750,562246,572453,592955,607816,614864,656448,658781,690422,708860)
> #Yearly times series
> sales_ts <- ts(sales,start=2000,end=2015,frequency=1)
> #Quaterly time series
> sales_ts <- ts(sales,start=2000,end=2015,frequency = 4)
> #Monthly time series
> sales_ts <- ts(sales,start = 2000,end=2015,frequency = 12)
> #ploting time series
> plot.ts(sales_ts)
>
```

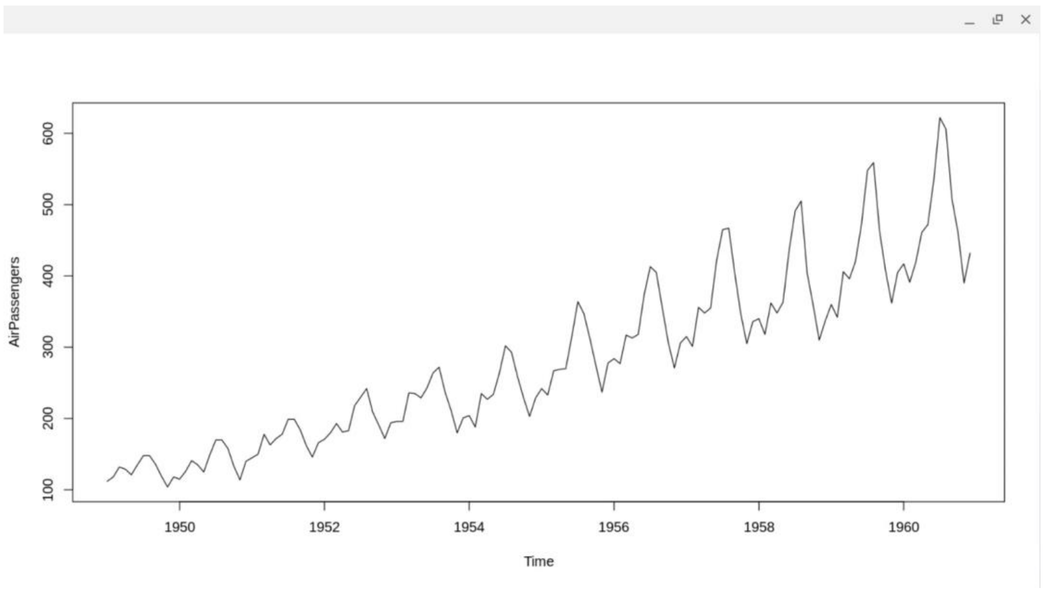


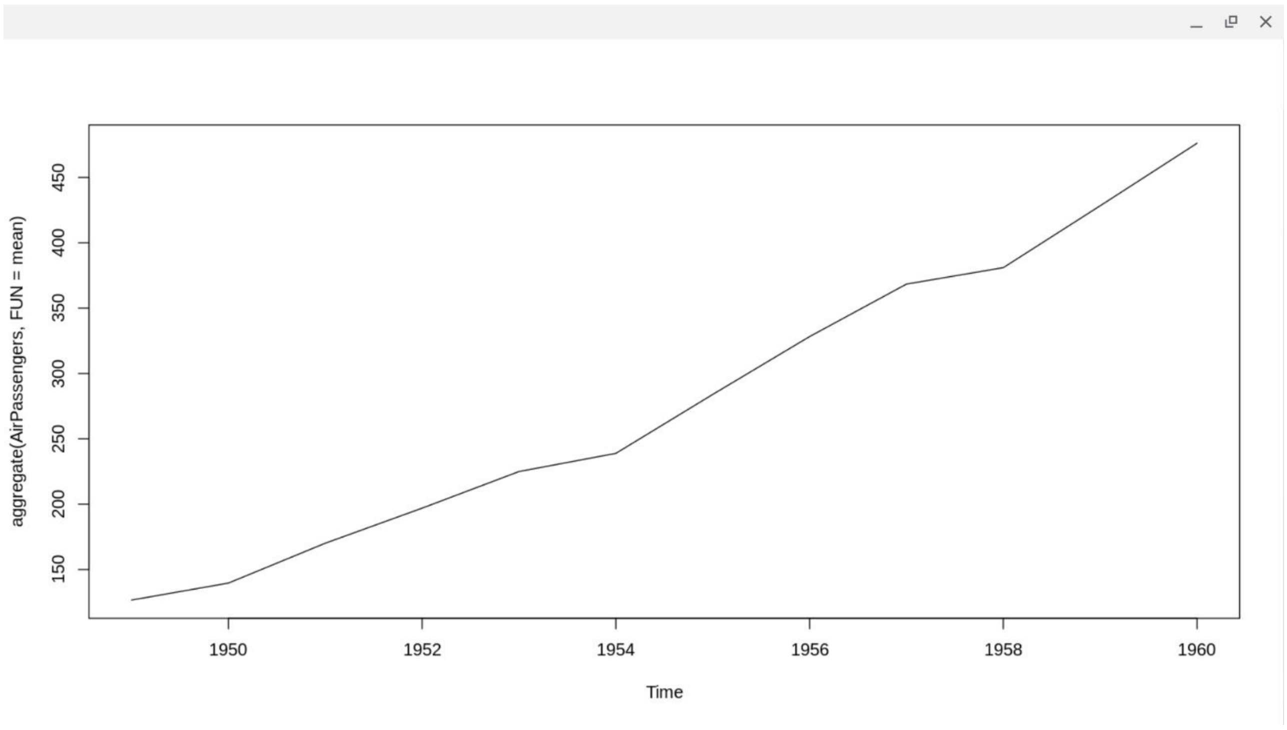
b.) Inbuilt Time Series

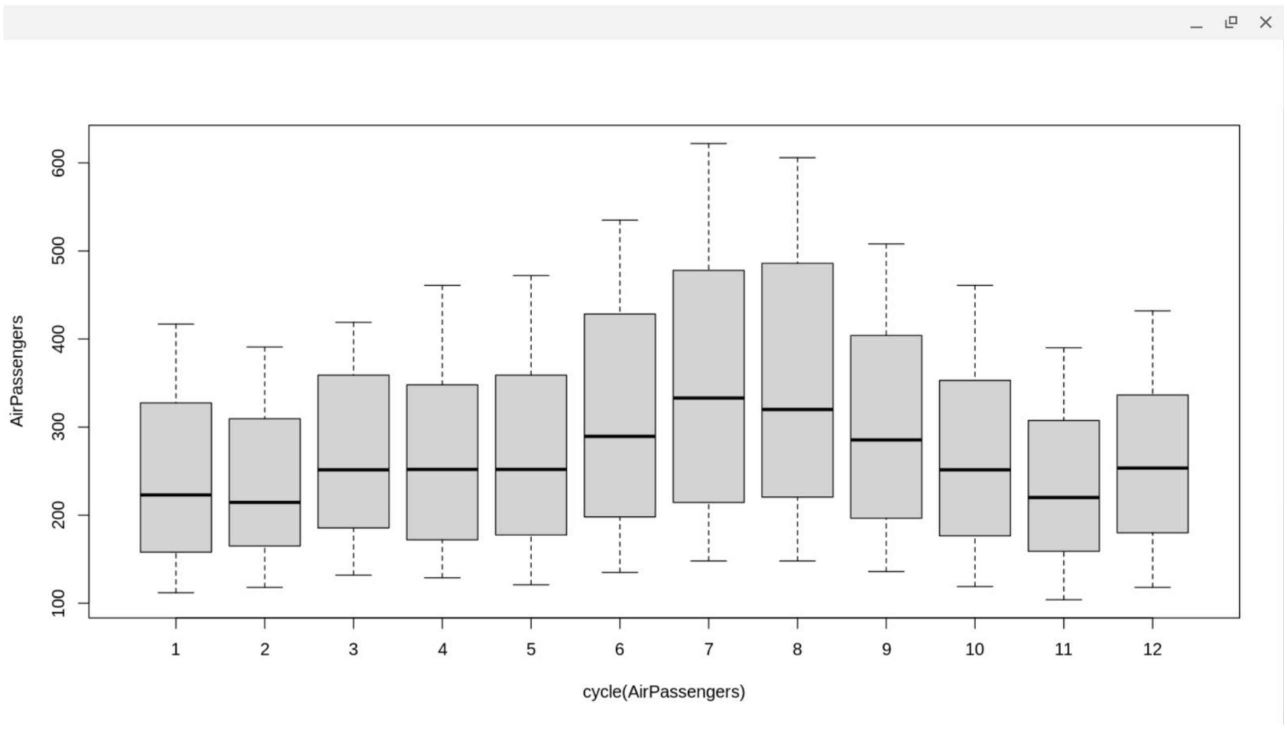
```
data("AirPassengers")
class(AirPassengers)
start(AirPassengers)
end(AirPassengers)
frequency(AirPassengers)
summary(AirPassengers)
plot(AirPassengers)
#plot the best fit line which can be used for regression
abline(reg=lm(AirPassengers ~ time(AirPassengers)))
#to print cycle across year
cycle(AirPassengers)
#to aggregate the cycle and display the trends as per year
plot(aggregate(AirPassengers,FUN = mean))
#to get Boxplot
boxplot(AirPassengers ~ cycle(AirPassengers))
```

Output:

```
> data("AirPassengers")
> class(AirPassengers)
[1] "ts"
> start(AirPassengers)
[1] 1949    1
> end(AirPassengers)
[1] 1960   12
> frequency(AirPassengers)
[1] 12
> summary(AirPassengers)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
> plot(AirPassengers)
> #plot the best fit line which can be used for regression
> abline(reg=lm(AirPassengers ~ time(AirPassengers)))
> #to print cycle across year
> 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
> #to aggregate the cycle and display the trends as per year
> plot(aggregate(AirPassengers,FUN = mean))
> #to get Boxplot
> boxplot(AirPassengers ~ cycle(AirPassengers))
```




aggregate(AirPassengers, FUN = mean)
450
400
350
300
250
200
150
1950
1952
1954
1956
1958
1960
Time


AirPassengers
600
500
400
300
200
100
1
2
3
4
5
6
7
8
9
10
11
12
cycle(AirPassengers)

Practical no: 6

Aim: Practical of Simple/Multiple Linear Regression

Description:

Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.

The key point in Simple Linear Regression is that the dependent variable must be a continuous/real value. However, the independent variable can be measured on continuous or categorical values.

Simple Linear regression algorithm has mainly two objectives:

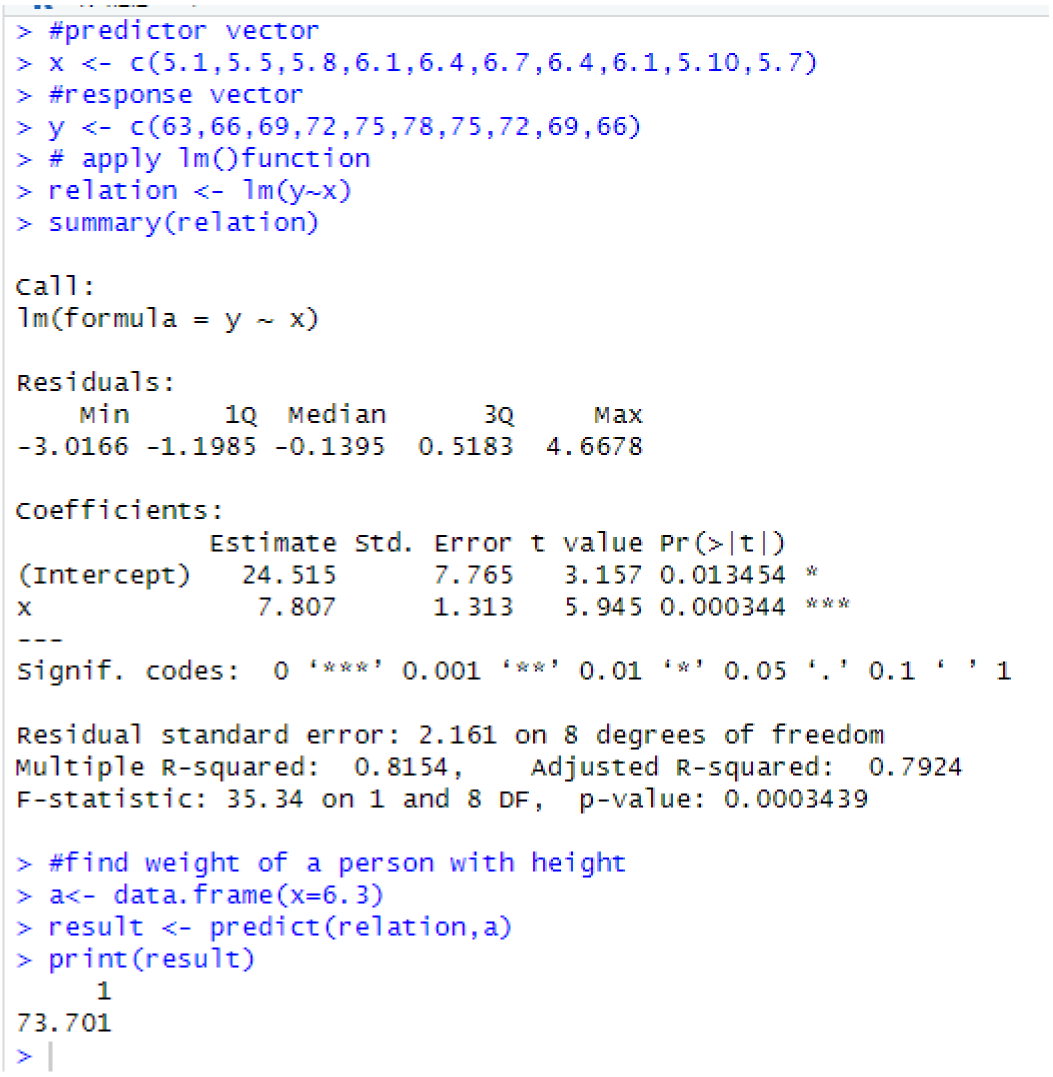
○ Model the relationship between the two variables. Such as the relationship between Income and expenditure, experience and Salary, etc.

○ Forecasting new observations. Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

Code:

```
#predictor vector

x <- c(5.1,5.5,5.8,6.1,6.4,6.7,6.4,6.1,5.10,5.7)

#response vector

y <- c(63,66,69,72,75,78,75,72,69,66)

# apply lm()function

relation <- lm(y~x)

summary(relation)

#find weight of a person with height

a<- data.frame(x=6.3)

result <- predict(relation,a)

print(result)
```

Output:



```
> #predictor vector
> x <- c(5.1,5.5,5.8,6.1,6.4,6.7,6.4,6.1,5.10,5.7)
> #response vector
> y <- c(63,66,69,72,75,78,75,72,69,66)
> # apply lm()function
> relation <- lm(y~x)
> summary(relation)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0166 -1.1985 -0.1395  0.5183  4.6678 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   24.515      7.765   3.157 0.013454 *  
x              7.807      1.313   5.945 0.000344 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.161 on 8 degrees of freedom
Multiple R-squared:  0.8154,    Adjusted R-squared:  0.7924 
F-statistic: 35.34 on 1 and 8 DF,  p-value: 0.0003439

> #find weight of a person with height
> a<- data.frame(x=6.3)
> result <- predict(relation,a)
> print(result)
     1 
73.701 
> 
```

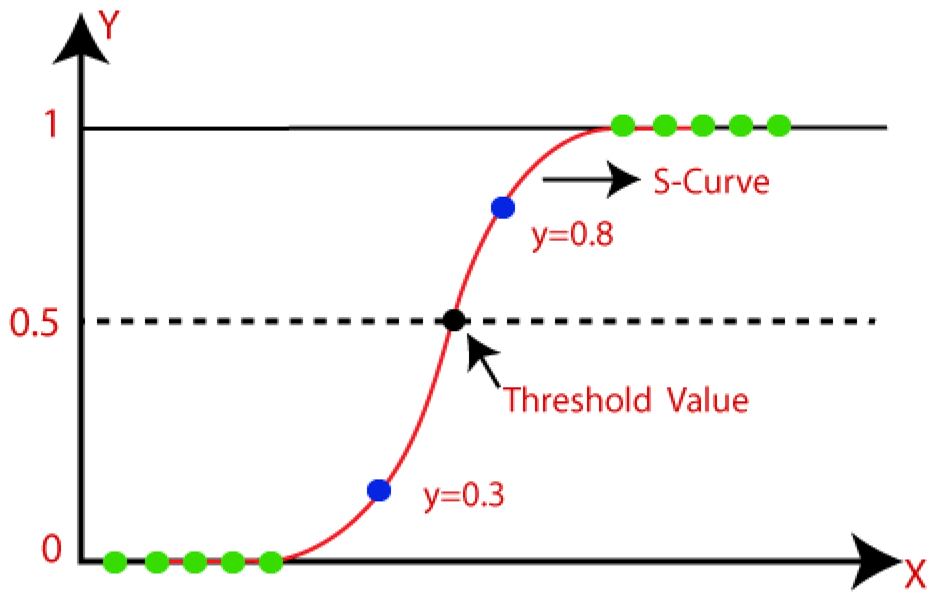
Practical 7

Aim: Practical of Logistic Regression

Description

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predictingthe categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be eitherYes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that howthey are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for theclassification. The below image is showing the logistic function:



Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predictedvalues to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot gobeyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.

- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multicollinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are givenbelow:

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of theequation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- Binomial: In binomial Logistic regression, there can be only two possible typesof the dependent variables, such as 0 or 1, Pass or Fail, etc.
- Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- Ordinal: In ordinal Logistic regression, there can be 3 or more possible orderedtypes of dependent variables, such as "low", "Medium", or "High".

Code:

lmmodel <- lm(mpg ~ hp +cyl+gear+wt,data=TrainData)

summary(lmmodel)

preds_new <- predict(lmmodel,newdata=TestData)

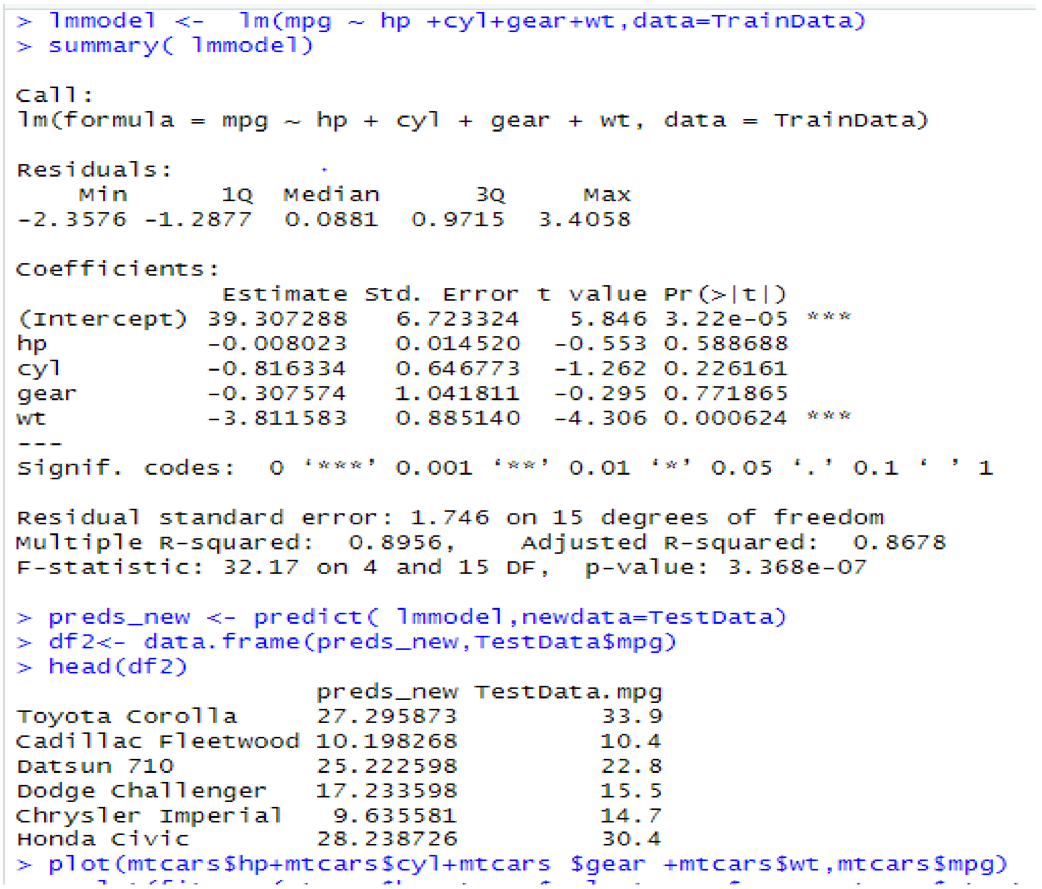
df2<- data.frame(preds_new,TestData$mpg)

head(df2)

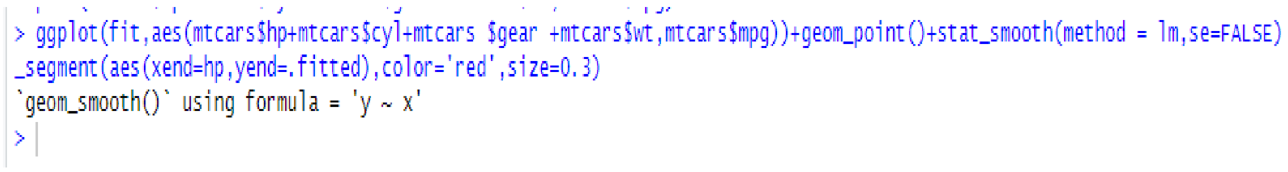
plot(mtcars$hp+mtcars$cyl+mtcars $gear +mtcars$wt,mtcars$mpg)

ggplot(fit,aes(mtcars$hp+mtcars$cyl+mtcars$gear +mtcars$wt,mtcars$mpg))+geom_point()+stat_smooth(method=lm,se=FALSE)+geom_segment(aes(xend=hp,yend=.fitted),color='red',size=0.3)

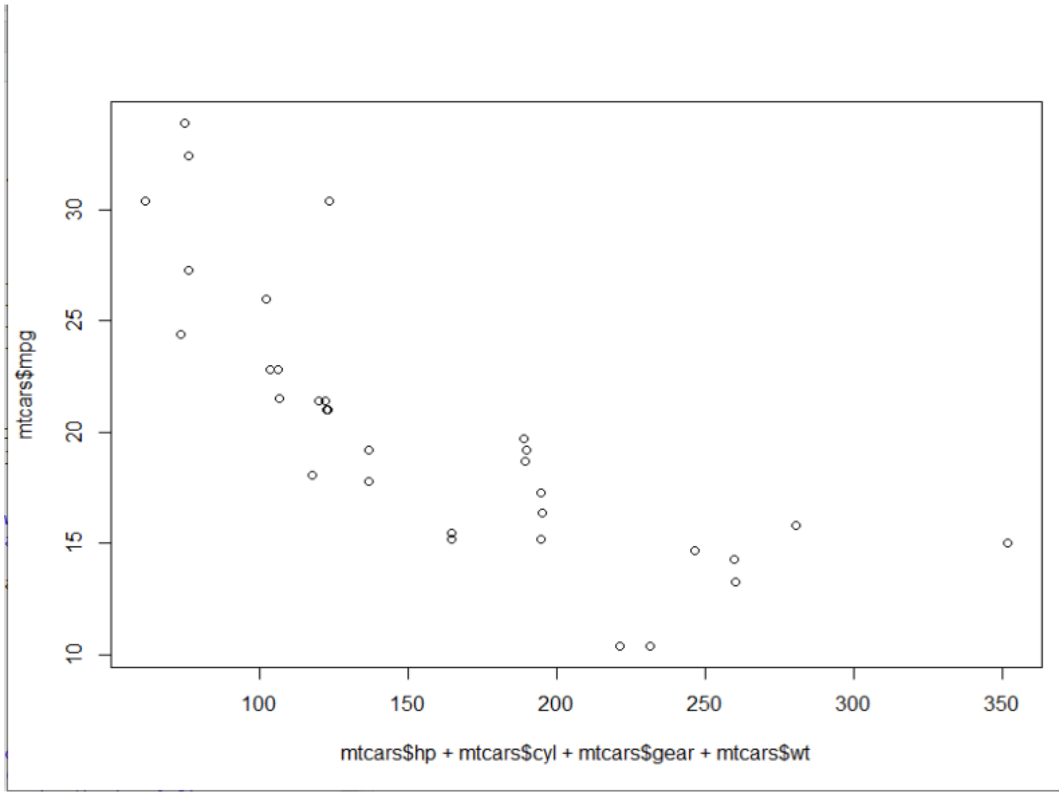
Output:



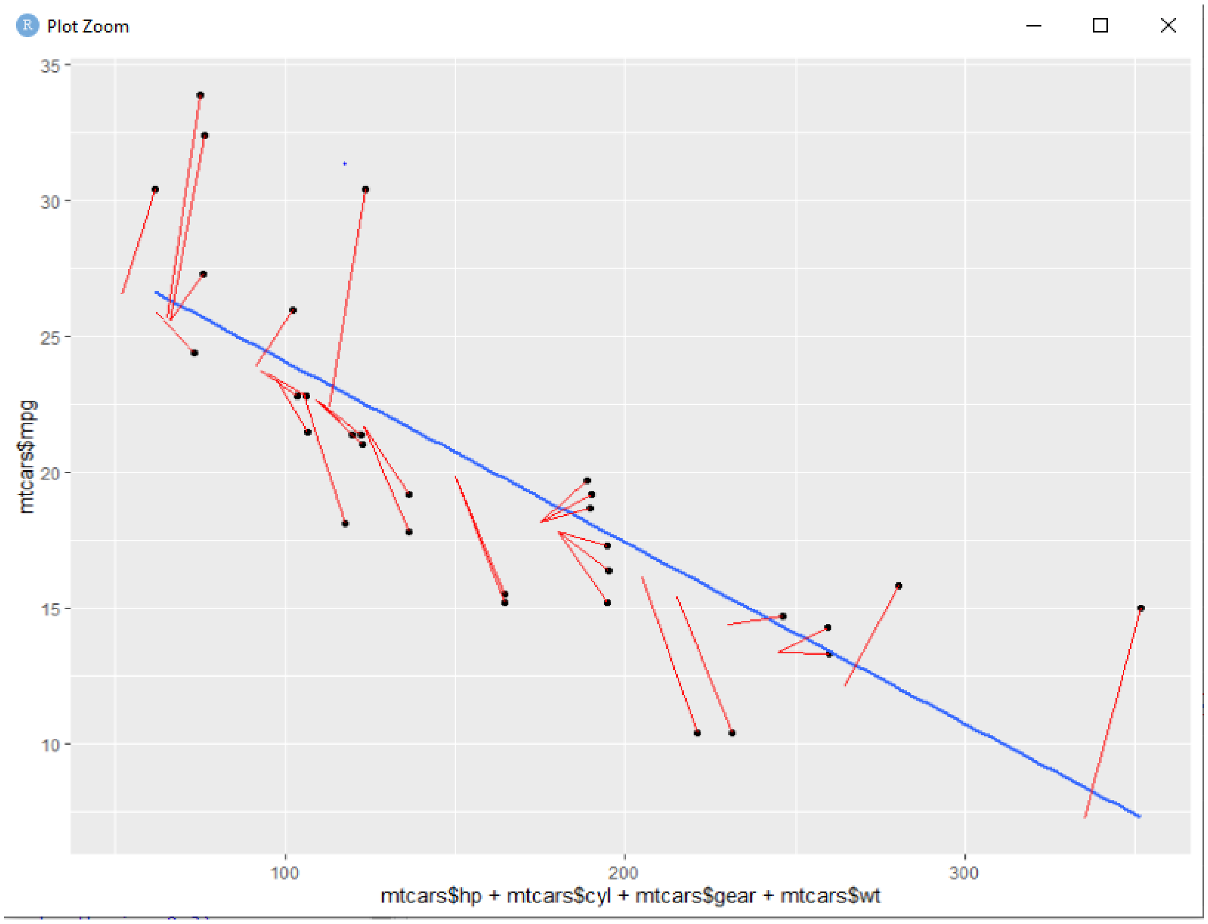
```
> lmmodel <-  lm(mpg ~ hp +cyl+gear+wt,data=TrainData)
> summary( lmmodel)

Call:
lm(formula = mpg ~ hp + cyl + gear + wt, data = TrainData)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.3576 -1.2877  0.0881  0.9715  3.4058 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 39.307288   6.723324   5.846 3.22e-05 ***
hp          -0.008023   0.014520  -0.553 0.588688    
cyl         -0.816334   0.646773  -1.262 0.226161    
gear        -0.307574   1.041811  -0.295 0.771865    
wt          -3.811583   0.885140  -4.306 0.000624 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.746 on 15 degrees of freedom
Multiple R-squared:  0.8956,	Adjusted R-squared:  0.8678 
F-statistic: 32.17 on 4 and 15 DF,  p-value: 3.368e-07

> preds_new <- predict( lmmodel,newdata=TestData)
> df2<- data.frame(preds_new,TestData$mpg)
> head(df2)
                   preds_new TestData.mpg
Toyota Corolla     27.295873         33.9
Cadillac Fleetwood 10.198268         10.4
Datsun 710         25.222598         22.8
Dodge Challenger   17.233598         15.5
Chrysler Imperial   9.635581         14.7
Honda Civic        28.238726         30.4
> plot(mtcars$hp+mtcars$cyl+mtcars $gear +mtcars$wt,mtcars$mpg)
```



```
> ggplot(fit,aes(mtcars$hp+mtcars$cyl+mtcars $gear +mtcars$wt,mtcars$mpg))+geom_point()+stat_smooth(method = lm,se=FALSE)
_segment(aes(xend=hp,yend=.fitted),color='red',size=0.3)
`geom_smooth()` using formula = 'y ~ x'
> 
```



mtcars$mpg
10
15
20
25
30
100
150
200
250
300
350
mtcars$hp + mtcars$cyl + mtcars$gear + mtcars$wt



Plot Zoom
mtcars$mpg
35
30
25
20
15
10
100
200
300
mtcars$hp + mtcars$cyl + mtcars$gear + mtcars$wt

Practical 8

Aim: Practical of Hypothesis Testing

Description

The hypothesis is defined as the supposition or proposed explanation based on insufficient evidence or assumptions. It is just a guess based on some known facts but has not yet been proven. A good hypothesis is testable, which results in either true or false.

○ Null Hypothesis: A null hypothesis is a type of statistical hypothesis which tells that there is no statistically significant effect exists in the given set of observations. It is also known as conjecture and is used in quantitative analysis to test theories about markets, investment, and finance to decide whether an idea is true or false.

○ Alternative Hypothesis: An alternative hypothesis is a direct contradiction of the null hypothesis, which means if one of the two hypotheses is true, then the other must be false. In other words, an alternative hypothesis is a type of statistical hypothesis which tells that there is some significant effect that exists in the given set of observations.

The significance level is the primary thing that must be set before starting an experiment. It is useful to define the tolerance of error and the level at which effect can be considered significantly. During the testing process in an experiment, a 95% significance level is accepted, and the remaining 5% can be neglected. The significance level also tells the critical or threshold value. For e.g., in an experiment, if the significance level is set to 98%, then the critical value is 0.02%.

The p-value in statistics is defined as the evidence against a null hypothesis. In other words, P-value is the probability that a random chance generated the data or something else that is equal or rarer under the null hypothesis condition.

If the p-value is smaller, the evidence will be stronger, and vice-versa which means the null hypothesis can be rejected in testing. It is always represented in a decimal form, such as 0.035.

Whenever a statistical test is carried out on the population and sample to find out P-value, then it always depends upon the critical value. If the p-value is less than the critical value, then it shows the effect is significant, and the null hypothesis can be rejected. Further, if it is higher than the critical value, it shows that there is no significant effect and hence fails to reject the Null Hypothesis.

Code:

a.)

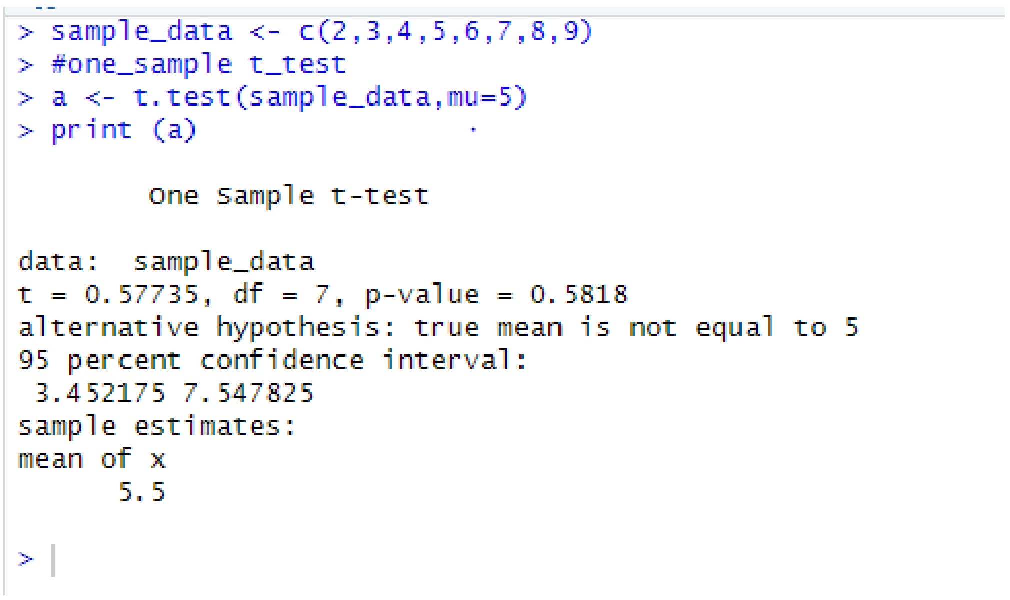
sample_data <- c(2,3,4,5,6,7,8,9)

#one_sample t_test

a <- t.test(sample_data,mu=5)

print (a)

Output:

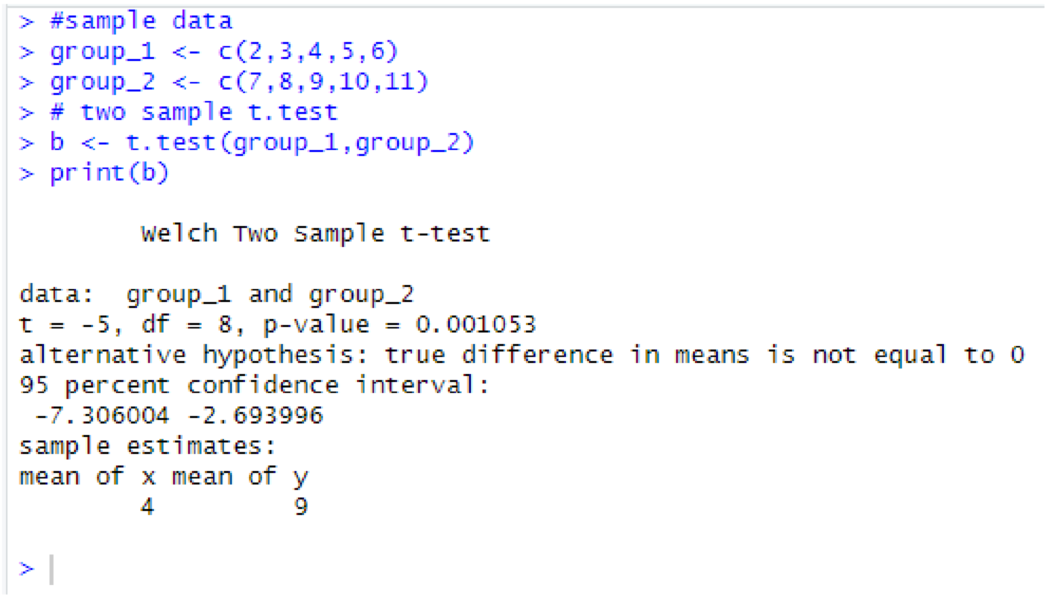


```
> sample_data <- c(2,3,4,5,6,7,8,9)
> #one_sample t_test
> a <- t.test(sample_data,mu=5)
> print (a)

	One Sample t-test

data:  sample_data
t = 0.57735, df = 7, p-value = 0.5818
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 3.452175 7.547825
sample estimates:
mean of x 
      5.5 

> 
```

b.)

```
#sample_data
group_1 <- c(2,3,4,5,6)
group_2 <- c(7,8,9,10,11)
# two sample t.test
b <- t.test(group_1,group_2)
print(b)
```

Output:



```
> #sample data
> group_1 <- c(2,3,4,5,6)
> group_2 <- c(7,8,9,10,11)
> # two sample t.test
> b <- t.test(group_1,group_2)
> print(b)

        Welch Two Sample t-test

data:  group_1 and group_2
t = -5, df = 8, p-value = 0.001053
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -7.306004 -2.693996
sample estimates:
mean of x mean of y 
        4         9 

> 
```

Practical 9

Aim: Practical on Analysis of Variance

Description:

Analysis of Variance Test, also known as ANOVA, to generalize the T-tests for multiple groups. Generally, we use the independent T-test in order to compare the means of thestate between two groups. We use ANOVA Test whenever we need a comparison of themeans of the state between more than two groups.

ANOVA test checks whether a difference in the average somewhere in the model ornot (checking whether there was an overall effect or not); however, this method doesn'ttell us the spot of the difference (if there is one). We can find the spot of the difference between the group by conducting the post hoc tests.

However, in order to perform any tests, we first have to define the null and alternate hypotheses:

1. Null Hypothesis:There is no noteworthy difference between the groups.
2. Alternate Hypothesis:There is a noteworthy difference between the groups.

We can perform an ANOVA Test by comparing two types of variations. The Firstvariation is between the sample means and the other one within each of the samples. The formula shown below describes one-way ANOVA Test statistics.

The output of the ANOVA formula, the F statistic (also known as the F-ratio), enables the analysis of the multiple sets of data in order to determine the variability among the samples and within samples.

We can write the formula for the One-way ANOVA test as illustrated below:

$$\boldsymbol{F} = \frac{Explained\ Variance}{Unexplained\ Variance}\ or\ \frac{Variance\ beween\ groups}{Variance\ within\ groups}$$

$$= \frac{Sum\ of\ squares\ between\ groups}{Sum\ of\ squares\ for\ error} = \frac{MST}{MSE}$$

$$= \frac{Mean\ squared\ error\ treatments}{Mean\ squared\ error} = \frac{SS_B/_{Df_T}}{SS_W/_{Df_E}} = \frac{SS_B/_{(k-1)}}{SS_W/_{(N-k)}}$$

Where,

$$SS_B = \sum_i n_i\,(\bar{y}_i - \bar{y})^2$$

$$SS_B = \sum_{ij} (\bar{y}_{ij} - \bar{y}_i)^2$$

Where,

y_i - Sample Mean in the ith group

n_i - Number of Observation in the ith groupy -

Total mean of the data

k - Total number of the groups

y_{ij} - jth observation in the out of k groupsN -

Overall sample size

Whenever we plot the ANOVA table, we can see all the above components in thefollowing format:

Source of Variation	**Sums of Squares (SS)**	**Degrees of Freedom (Df)**	**Mean Squares (MS)**	**F**
Between Treatments	SS_B	$k - 1$	$MST = SS_B/(k - 1)$	$F = \frac{MST}{MSE}$
Error (or Residual)	SS_E	$N - k$	$MSE = SS_B/(N - k)$	
Total	$SS_T = SS_B + SS_E$	$N - 1$	T	

Usually, if the p-value belonging to the F is smaller than 0.05, then the null hypothesis is excluded, and the alternative hypothesis is maintained. In the case of the null hypothesis rejection, we can say that the means of all the sets/groups aren't equal.

ANOVA Test Assumptions

Before performing an ANOVA test, we must make certain assumptions, as shown below:

1. We can obtain observations randomly and independently from the population defined by the factor levels.

2. The data for every level of the factor is distributed generally.

3. Case Independent: The sample cases must be independent of each other.

4. Variance Homogeneity: Homogeneity signifies that the variance between thegroup needs to be around equal.

ANOVA Tests can be classified into three major types. These types are shown below:

1. <u>One-Way ANOVA Test</u>

An Analysis of Variance Test that has only one independent variable is known as the One-way ANOVA Test. For instance, a country can assess the differences in the cases of Coronavirus, and a Country can have multiple categories for comparison.

2. Two-Way ANOVA Test

An Analysis of Variance Test that has two independent variables is known as a Two-way ANOVA test. This test is also known as Factorial ANOVA Test.

For example, expanding the above example, a two-way ANOVA can examine the difference in the cases of Coronavirus (the dependent variable) by Age Group (the first independent variable) and Gender (the second independent variable). The two-way ANOVA can be utilized in order to examine the interaction among these two independent variables. Interactions denote that the differences are uneven across all classes of the independent variables.

Suppose that the old age group may have higher cases of Coronavirus overall compared to the young age group; however, this difference could vary in countries in Europe compared to countries in Asia.

3. n-Way ANOVA Test

An Analysis of Variance Test is considered an n-way ANOVA Test if a researcher uses more than two independent variables. Here n represents the number of independent variables we have. This Test is also known as MANOVA Test.

For example, we can examine potential differences in cases of Coronavirus using independent variables like Country, Age group, Gender, Ethnicity, and a lot more simultaneously.

Code:

a.)

#Generate 3 sets of data

x <- rnorm(50, mean = 10, sd=2)

y <- rnorm(50, mean = 12, sd=2)

z <- rnorm(50, mean = 14, sd=2)

#Combine the data into single frame

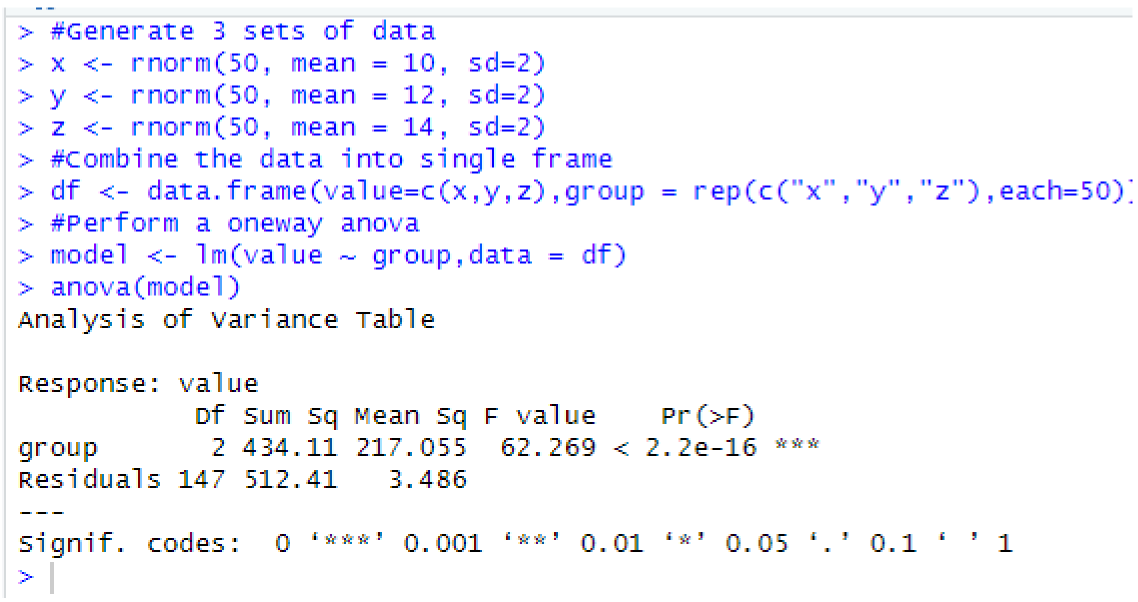
df <- data.frame(value=c(x,y,z),group = rep(c("x","y","z"),each=50))

#Perform a oneway anova

model <- lm(value ~ group,data = df)

anova(model)

Output:



```
> #Generate 3 sets of data
> x <- rnorm(50, mean = 10, sd=2)
> y <- rnorm(50, mean = 12, sd=2)
> z <- rnorm(50, mean = 14, sd=2)
> #Combine the data into single frame
> df <- data.frame(value=c(x,y,z),group = rep(c("x","y","z"),each=50)
> #Perform a oneway anova
> model <- lm(value ~ group,data = df)
> anova(model)
Analysis of Variance Table

Response: value
           Df Sum Sq Mean Sq F value    Pr(>F)    
group       2 434.11 217.055  62.269 < 2.2e-16 ***
Residuals 147 512.41   3.486                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

b.) Anova Testing on Inbuilt dataset

Code:

#load mtcars dataset

data(mtcars)

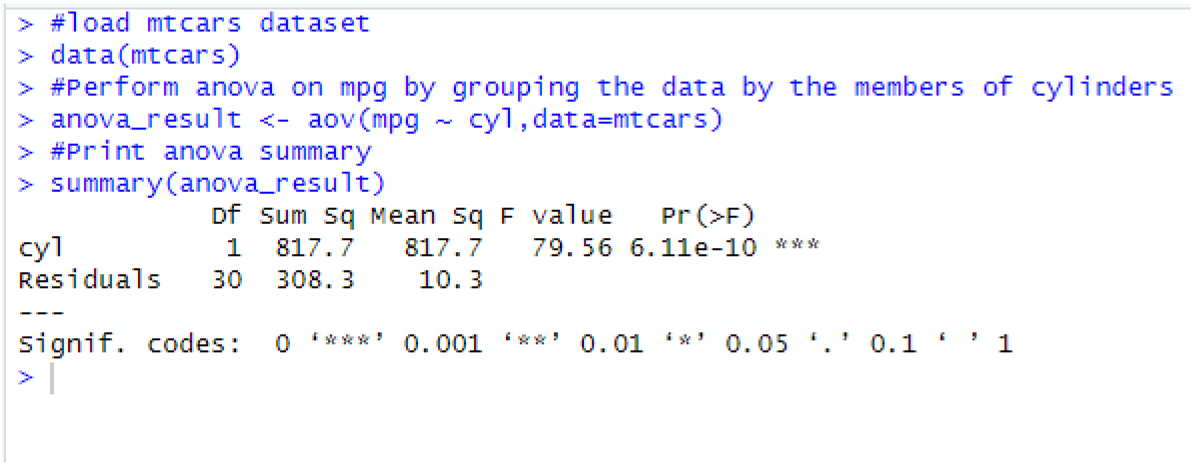
#Perform anova on mpg by grouping the data by the members of cylinders

anova_result <- aov(mpg ~ cyl,data=mtcars)

#Print anova summary

summary(anova_result)

Output:



```
> #load mtcars dataset
> data(mtcars)
> #Perform anova on mpg by grouping the data by the members of cylinders
> anova_result <- aov(mpg ~ cyl,data=mtcars)
> #Print anova summary
> summary(anova_result)
            Df Sum Sq Mean Sq F value   Pr(>F)    
cyl          1  817.7   817.7   79.56 6.11e-10 ***
Residuals   30  308.3    10.3                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

c.)

Code:

#Create dataframe with the data

data_1 <- data.frame(group=c(rep("A",5),rep("B",5),rep("C",5)),value=c(10,12,15,18,20,8,10,12,14,16,5,8,10,12,14))

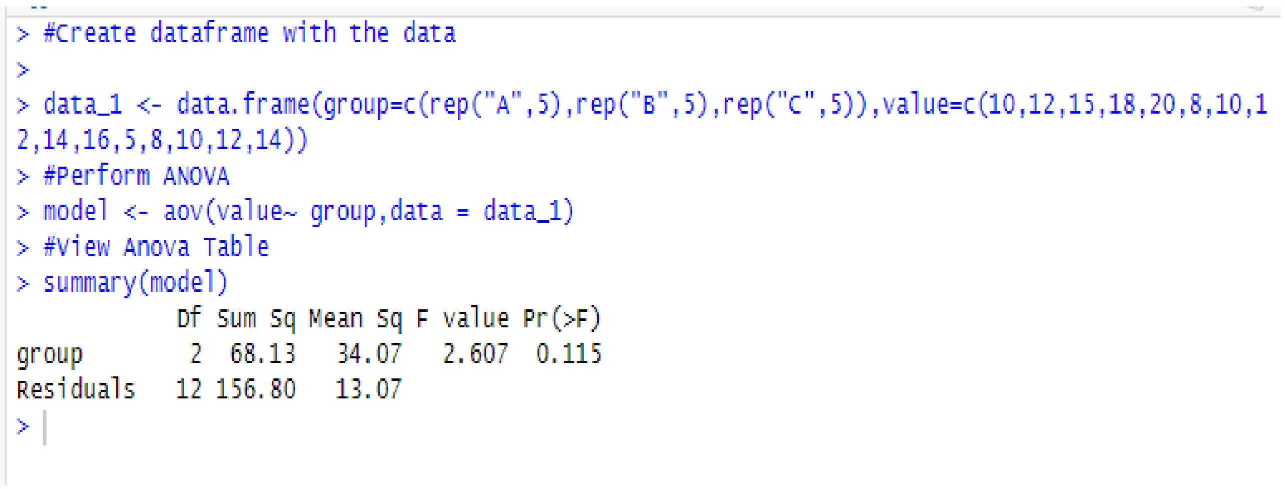
#Perform ANOVA

model <- aov(value~ group,data = data_1)

#View Anova Table

summary(model)

Output:



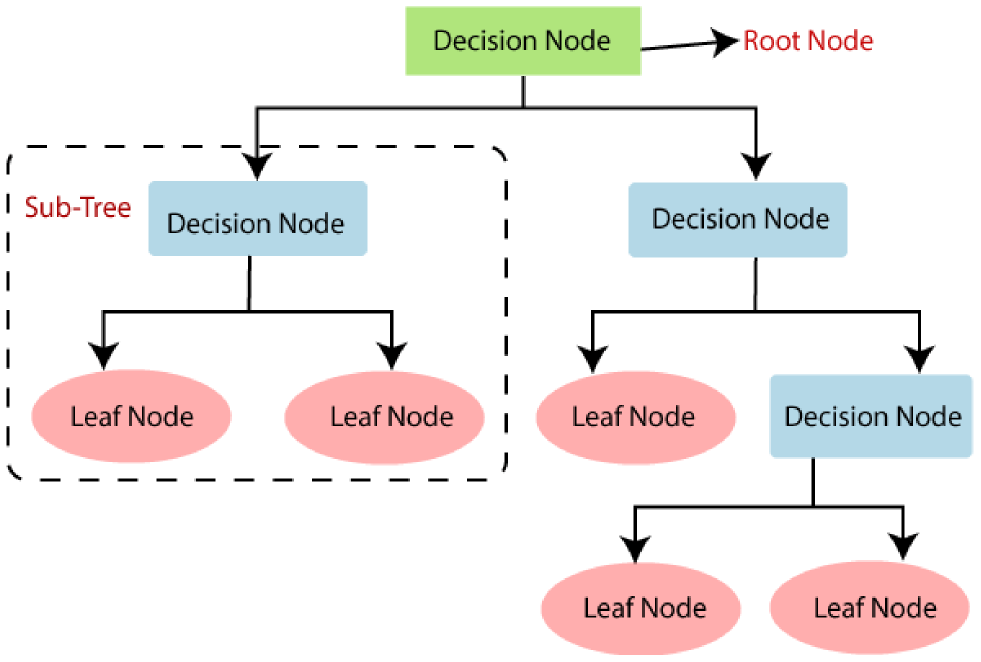
```
> #Create dataframe with the data
>
> data_1 <- data.frame(group=c(rep("A",5),rep("B",5),rep("C",5)),value=c(10,12,15,18,20,8,10,1
2,14,16,5,8,10,12,14))
> #Perform ANOVA
> model <- aov(value~ group,data = data_1)
> #View Anova Table
> summary(model)
            Df Sum Sq Mean Sq F value Pr(>F)
group        2  68.13   34.07   2.607  0.115
Residuals   12 156.80   13.07
> 
```

Practical 10

Aim: Practical of Decision Tree

Description

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules andeach leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given Dataset.
- It is a graphical representation for getting all the possible solutions to aproblem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



Decision Tree Terminologies:

Root Node: Root node is from where the decision tree starts. It represents the entiredataset, which further gets divided into two or more homogeneous sets.

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregatedfurther after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node intosub-nodes according to the given conditions.

Branch/Sub Tree: A tree formed by splitting the tree.

Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodesare called the child nodes.

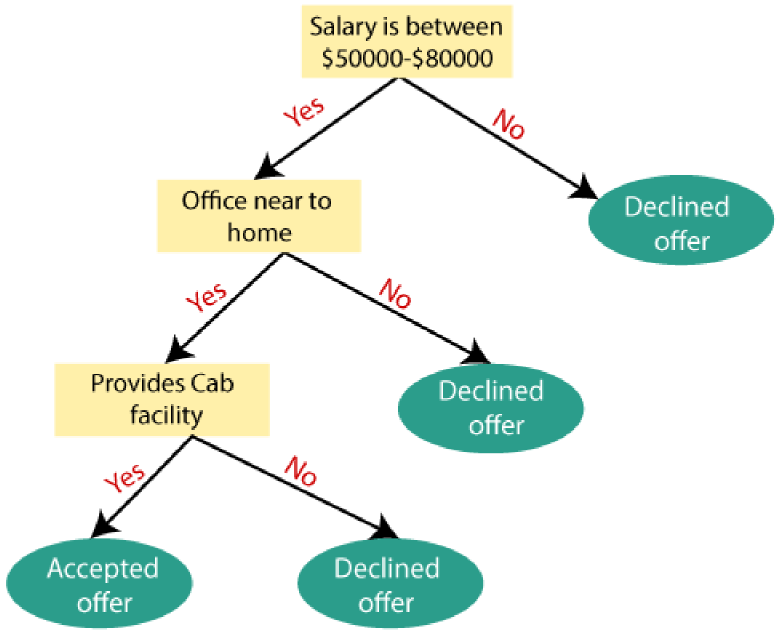
In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the completedataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure(ASM).
- Step-3: Divide the S into subsets that contains possible values for the bestattributes.
- Step-4: Generate the decision tree node, which contains the best attribute.

- Step-5: Recursively make new decision trees using the subsets of the datasetcreated in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. Thereare two popular techniques for ASM, which are:

- Information Gain
- Gini Index

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. Itcan be calculated using the below formula:

  Information Gain= Entropy(S) - [(Weighted Avg) *Entropy (each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:
Entropy(s) = -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no

2. Gini Index

- Gini index is a measure of impurity or purity used while creating a decision treein the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the highGini index.
- It only creates binary splits, and the CART algorithm uses the Gini index tocreate binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

<u>Pruning:</u>

Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

- Cost Complexity Pruning
- Reduced Error Pruning.

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human followwhile making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

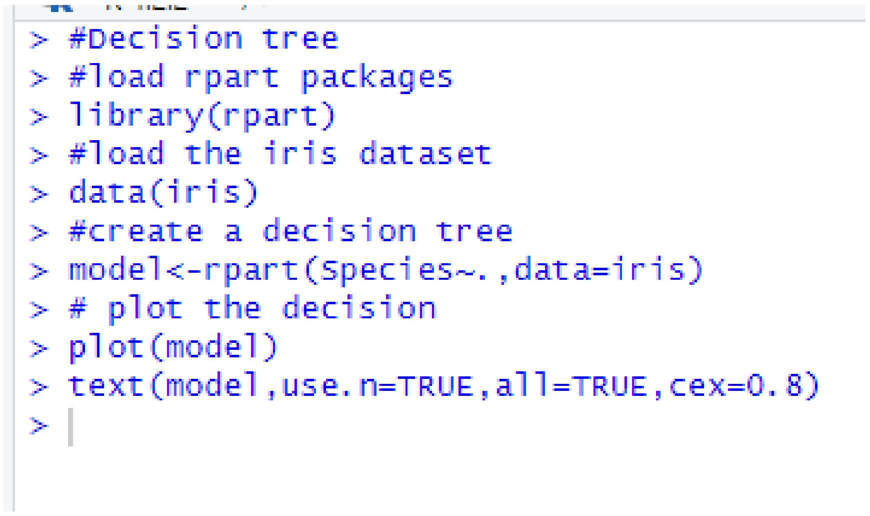
- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

For more class labels, the computational complexity of the decision tree mayincrease

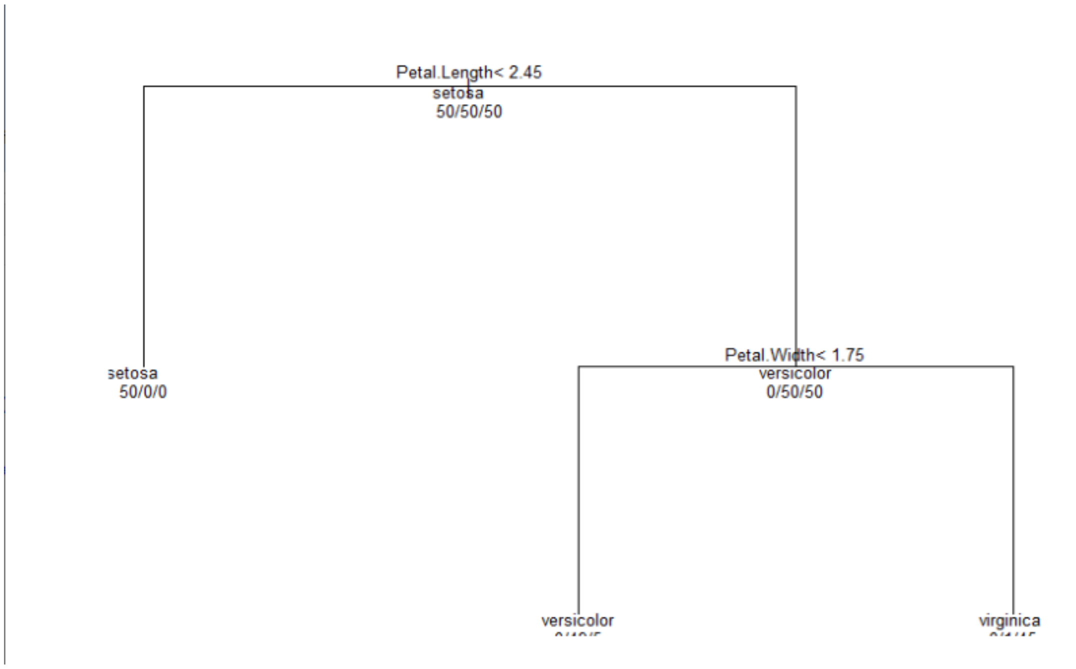
Code:

```
#Decision tree
#load rpart packages
library(rpart)
#load the iris dataset
data(iris)
#create a decision tree
model<-rpart(Species~.,data=iris)
# plot the decision
plot(model)
text(model,use.n=TRUE,all=TRUE,cex=0.8)
```

Output:



```
> #Decision tree
> #load rpart packages
> library(rpart)
> #load the iris dataset
> data(iris)
> #create a decision tree
> model<-rpart(Species~.,data=iris)
> # plot the decision
> plot(model)
> text(model,use.n=TRUE,all=TRUE,cex=0.8)
> 
```


Petal.Length< 2.45
setosa
50/50/50
setosa
50/0/0
Petal.Width< 1.75
versicolor
0/50/50
versicolor
virginica