INDEX

Subject: Data Science

Sr. No	Practical	Date	Page no.	Remark
1	Practical of Data collection, Data curation and management for Unstructured data (NoSQL)			
2	Practical of Data collection, Data curation and management for Large-scale Data system (such as MongoDB)			
3	Practical of Principal Component Analysis			
4	Practical of Clustering			
5	Practical of Time-series forecasting			
6	Practical of Simple/Multiple Linear Regression			
7	Practical of Logistics Regression			
8	Practical of Hypothesis testing			
9	Practical of Analysis of Variance			
10	Practical of Decision Tree			

PRACTICAL 1: Practical of Data collection, Data curation and management for Unstructured data (NoSQL)

Q. Create a NoSQL data for storing information of students where for 2 students name and course is known, for 3 students name, course and mobile number is known, for other 2 students name and address is known. View same data.
Retrieve all data stored in student.
Retrieve information for specified student's name
Retrieve the data of only specified keys satisfying a given condition like course = CS

Note: From C:\Program Files\MongoDB\Server\4.0\bin
* first start mongod.exe and then mongo.exe

1. create NoSQl data for storing information:

```
> use student;
switched to db student
> db.student.insertMany([{name:'rucha',course:'CS'},{name:'anushka',course:'IT'},{name:'tanisha',course:'CS',
mobile:9807454609},{name:'prachi',course:'IT',mobile:9820424509},{name:'mahima',course:'Bcom', mobile:9167855
097},{name:'bhakti',address:'mulund'},{name:'anisha',address:'kalyan'}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5c9c4e287c3da483585299fd"),
                ObjectId("5c9c4e287c3da483585299fe"),
                ObjectId("5c9c4e287c3da483585299ff"),
                ObjectId("5c9c4e287c3da48358529a00"),
                ObjectId("5c9c4e287c3da48358529a01"),
                ObjectId("5c9c4e287c3da48358529a02"),
                ObjectId("5c9c4e287c3da48358529a03")
        ]
}
```

2. Retrieve all data stored in student.

```
> db.student.find()
{ "_id" : ObjectId("5c9c4e287c3da483585299fd"), "name" : "rucha", "course" : "CS" }
{ "_id" : ObjectId("5c9c4e287c3da483585299fe"), "name" : "anushka", "course" : "IT" }
{ "_id" : ObjectId("5c9c4e287c3da483585299ff"), "name" : "tanisha", "course" : "CS", "mobile" : 9807454609 }
{ "_id" : ObjectId("5c9c4e287c3da48358529a00"), "name" : "prachi", "course" : "IT", "mobile" : 9820424509 }
{ "_id" : ObjectId("5c9c4e287c3da48358529a01"), "name" : "mahima", "course" : "Bcom", "mobile" : 9167855097 }
{ "_id" : ObjectId("5c9c4e287c3da48358529a02"), "name" : "bhakti", "address" : "mulund" }
{ "_id" : ObjectId("5c9c4e287c3da48358529a03"), "name" : "anisha", "address" : "kalyan" }
```

3. Retrieve information for specified student's name

```
> db.student.find({name:'rucha'})
{ "_id" : ObjectId("5c9c4e287c3da483585299fd"), "name" : "rucha", "course" : "CS" }
```

4. Retrieve the data of only specified keys satisfying a given condition like course = CS
   Here we only want name and not id hence _id:0

```
> db.student.find({course:'CS'},{_id:0,name:1})
{ "name" : "rucha" }
{ "name" : "tanisha" }
```

PRACTICAL 2: Practical of Data collection, Data curation and management for Large-scale Data system (such as MongoDB)

Note: Download MongoDB 4.0 and from C:\Program Files\MongoDB\Server\4.0\bin
* first start mongod.exe and then mongo.exe

1)To Show database in mongoDB

```
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
student  0.000GB
```

2) To Create new database in mongoDB

```
> use newdb;
switched to db newdb
```

3) To know which database in use

```
> db;
newdb
```

4) use name of database

```
> use admin;
switched to db admin
> use newdb;
switched to db newdb
```

5) For inserting record
Note:-you don't need to create collection. MongoDB creates collection automatically, when you insert some document.
Syntax:- db.collection_name.insert(document)

i)Insert one record

```
> db.mongodb.insert({"1":"Rucha"});
WriteResult({ "nInserted" : 1 })
```

ii) Insert many record

```
> db.people.insertMany([{name:"Tanisha", age:20},{name:"anushka", age:21}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5c9c54567c3da48358529a05"),
                ObjectId("5c9c54567c3da48358529a06")
        ]
}
```

6) To display collection

i) Normal display

```
> db.people.find();
{ "_id" : ObjectId("5c9c54567c3da48358529a05"), "name" : "Tanisha", "age" : 20 }
{ "_id" : ObjectId("5c9c54567c3da48358529a06"), "name" : "anushka", "age" : 21 }
```

ii)Display in proper format

```
> db.people.find().pretty();
{
        "_id" : ObjectId("5c9c54567c3da48358529a05"),
        "name" : "Tanisha",
        "age" : 20
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
```

7)Find Specific record

Note:-Use function db.collection.findOne(document) if there are many records

```
> db.people.find({name:'Tanisha'}).pretty();
{
        "_id" : ObjectId("5c9c54567c3da48358529a05"),
        "name" : "Tanisha",
        "age" : 20
}
```

8)Sorting data in mongoDB

i)Ascending order

```
> db.people.find().sort({age:1}).pretty();
{
        "_id" : ObjectId("5c9c54567c3da48358529a05"),
        "name" : "Tanisha",
        "age" : 20
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a09"),
        "name" : "Rucha",
        "age" : 20
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a07"),
        "name" : "Prachi",
        "age" : 22
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a08"),
        "name" : "Priya",
        "age" : 23
}
```

i)descending order

```
> db.people.find().sort({age:-1}).pretty();
{
        "_id" : ObjectId("5c9c55457c3da48358529a08"),
        "name" : "Priya",
        "age" : 23
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a07"),
        "name" : "Prachi",
        "age" : 22
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a05"),
        "name" : "Tanisha",
        "age" : 20
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a09"),
        "name" : "Rucha",
        "age" : 20
}
```

iii) Sorting data in mongoDB with limit

```
> db.people.find().sort({age:-1}).limit(3).pretty();
{
        "_id" : ObjectId("5c9c55457c3da48358529a08"),
        "name" : "Priya",
        "age" : 23
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a07"),
        "name" : "Prachi",
        "age" : 22
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
```

9) Update in mongoDB

i) update one record

Note:-If more than one record with same name then it will modify first one

```
> db.people.update({name:"Tanisha"},{name:"Tanisha",age:21});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.people.find().pretty();
{
        "_id" : ObjectId("5c9c54567c3da48358529a05"),
        "name" : "Tanisha",
        "age" : 21
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a07"),
        "name" : "Prachi",
        "age" : 22
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a08"),
        "name" : "Priya",
        "age" : 23
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a09"),
        "name" : "Rucha",
        "age" : 20
}
```

i) update many records

```
> db.people.updateMany({name:"Tanisha"},{$set:{name:"Tanisha",age:15}});
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
> db.people.find().pretty();
{
        "_id" : ObjectId("5c9c54567c3da48358529a05"),
        "name" : "Tanisha",
        "age" : 15
}
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a07"),
        "name" : "Prachi",
        "age" : 22
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a08"),
        "name" : "Priya",
        "age" : 23
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a09"),
        "name" : "Rucha",
        "age" : 20
}
{
        "_id" : ObjectId("5c9c563f7c3da48358529a0a"),
        "name" : "Tanisha",
        "age" : 15
}
```

10)Show collections

```
> show collections;
mongodb
people
```

11) Delete record in mongoDB

```
> db.people.deleteOne({name:"Tanisha"});
{ "acknowledged" : true, "deletedCount" : 1 }
> db.people.find().pretty();
{
        "_id" : ObjectId("5c9c54567c3da48358529a06"),
        "name" : "anushka",
        "age" : 21
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a07"),
        "name" : "Prachi",
        "age" : 22
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a08"),
        "name" : "Priya",
        "age" : 23
}
{
        "_id" : ObjectId("5c9c55457c3da48358529a09"),
        "name" : "Rucha",
        "age" : 20
}
{
        "_id" : ObjectId("5c9c563f7c3da48358529a0a"),
        "name" : "Tanisha",
        "age" : 15
}
```

Note:-Try with remove function it will give same result as deleteOne

```
> db.people.remove({name:"Tanisha"});
WriteResult({ "nRemoved" : 1 })
```

Note:- remove without parameter will remove all record

```
> db.people.remove({});
WriteResult({ "nRemoved" : 4 })
```

11)Drop Function in mongoDB

```
> db.people.drop();
true
```

Basics of R Studio

Data Set:

	A	B	C	D	E	F	G
1	emp_id	first_name	last_name	grade	loc	dept	salary
2	1	Rucha	Mahabal	A	Mumbai	CS	80000
3	2	Anushka	Bommakanty	B	Mumbai	IT	50000
4	3	Prachi	Sawant	A	Mumbai	HR	60000
5	4	Tanisha	Ramani	A	Pune	CS	40000
6	5	Mahima	Prajapati	B	Nasik	HR	30000
7	6	Anisha	Roy	C	Bangalore	IT	25000
8	7	Komal	Tripathi	C	Pune	IT	25000
9	8	Bhakti	Somaiya	B	Bangalore	CS	28000
10	9	Priya	Kulkarni	A	Mumbai	IT	

Q.1) Creating Object of data value

```
> emp <- read.csv(file.choose())
>
```

Object 'emp' Created

Q.2) Printing Data value

emp

Output:-

```
> emp
  emp_id first_name  last_name grade       loc dept salary
1      1      Rucha    Mahabal     A    Mumbai   CS  80000
2      2    Anushka  Bommakanty     B    Mumbai   IT  50000
3      3     Prachi     Sawant     A    Mumbai   HR  60000
4      4    Tanisha     Ramani     A      Pune   CS  40000
5      5     Mahima  Prajapati     B     Nasik   HR  30000
6      6     Anisha        Roy     C Bangalore   IT  25000
7      7      Komal   Tripathi     C      Pune   IT  25000
8      8     Bhakti    Somaiya     B Bangalore   CS  28000
9      9      Priya   Kulkarni     A    Mumbai   IT     NA
```

Q.3) Summary of data value

summary(emp)

Output:-

```
> summary(emp)
     emp_id     first_name      last_name grade        loc    dept
 Min.   :1   Anisha  :1    Bommakanty:1   A:4   Bangalore:2   CS:3
 1st Qu.:3   Anushka :1    Kulkarni  :1   B:3   Mumbai   :4   HR:2
 Median :5   Bhakti  :1    Mahabal   :1   C:2   Nasik    :1   IT:4
 Mean   :5   Komal   :1    Prajapati :1         Pune     :2
 3rd Qu.:7   Mahima  :1    Ramani    :1
 Max.   :9   Prachi  :1    Roy       :1
             (Other) :3    (Other)   :3
     salary
 Min.   :25000
 1st Qu.:27250
 Median :35000
 Mean   :42250
 3rd Qu.:52500
 Max.   :80000
 NA's   :1
```

Q.4) Performing sum on data value

Note: As one of the record has salary as NA output is NA

```
> salary_total <- sum(emp$salary)
> salary_total
[1] NA
```

To get the output:

```
> salary_total <- sum((emp$salary), na.rm = TRUE)
> salary_total
[1] 338000
```

Q.5)Display First ,last few record from data value

i) head(emp)- for first few record

Output:-

```
> head(emp)
  emp_id first_name  last_name grade       loc dept salary
1      1      Rucha    Mahabal     A    Mumbai   CS  80000
2      2   Anushka  Bommakanty     B    Mumbai   IT  50000
3      3    Prachi      Sawant     A    Mumbai   HR  60000
4      4   Tanisha      Ramani     A      Pune   CS  40000
5      5    Mahima  Prajapati     B     Nasik   HR  30000
6      6    Anisha        Roy     C Bangalore   IT  25000
```

ii) tail(emp)-for last few record

Output:-

```
> tail(emp)
  emp_id first_name last_name grade       loc dept salary
4      4   Tanisha     Ramani     A      Pune   CS  40000
5      5    Mahima Prajapati     B     Nasik   HR  30000
6      6    Anisha       Roy     C Bangalore   IT  25000
7      7     Komal  Tripathi     C      Pune   IT  25000
8      8    Bhakti   Somaiya     B Bangalore   CS  28000
9      9     Priya  Kulkarni     A    Mumbai   IT     NA
```

Q.6)Finding Mean of data(to specific column)

Output:-

```
> mean_salary <- mean((emp$salary), na.rm = TRUE)
> mean_salary
[1] 42250
```

Q.7)For condition like salary >5000

Output:-

```
> emp_data <- subset(emp, loc=="Mumbai" & salary>30000)
> emp_data
  emp_id first_name  last_name grade    loc dept salary
1      1      Rucha    Mahabal     A Mumbai   CS  80000
2      2   Anushka  Bommakanty     B Mumbai   IT  50000
3      3    Prachi      Sawant     A Mumbai   HR  60000
> grade_location <- subset(emp, !(loc == "Mumbai") & (grade=="A"))
> grade_location
  emp_id first_name last_name grade  loc dept salary
4      4   Tanisha     Ramani     A Pune   CS  40000
```

Q.8) For displaying particular record

i) Output:-

```
> emp[,c(1,2,3)]
  emp_id first_name  last_name
1      1      Rucha    Mahabal
2      2   Anushka  Bommakanty
3      3    Prachi      Sawant
4      4   Tanisha      Ramani
5      5    Mahima  Prajapati
6      6    Anisha        Roy
7      7     Komal   Tripathi
8      8    Bhakti    Somaiya
9      9     Priya   Kulkarni
```

ii)Output:-

```
> emp[c(5:10),c(1,2,3)]
   emp_id first_name last_name
5       5    Mahima Prajapati
6       6    Anisha       Roy
7       7     Komal  Tripathi
8       8    Bhakti   Somaiya
9       9     Priya  Kulkarni
NA     NA      <NA>      <NA>
```

Q.9) For display data in order

i) For ascending order

Output:-

```
> sort_salary_asc <- emp[order(emp$salary),]
> sort_salary_asc
  emp_id first_name  last_name grade       loc dept salary
6      6     Anisha        Roy     C Bangalore   IT  25000
7      7      Komal   Tripathi     C      Pune   IT  25000
8      8     Bhakti    Somaiya     B Bangalore   CS  28000
5      5     Mahima  Prajapati     B     Nasik   HR  30000
4      4   Tanisha      Ramani     A      Pune   CS  40000
2      2   Anushka  Bommakanty     B    Mumbai   IT  50000
3      3    Prachi      Sawant     A    Mumbai   HR  60000
1      1     Rucha    Mahabal     A    Mumbai   CS  80000
9      9     Priya   Kulkarni     A    Mumbai   IT     NA
```

ii) For descending order

Output:-

```
> sort_salary_desc <- emp[order(-emp$salary),]
> sort_salary_desc
  emp_id first_name  last_name grade       loc dept salary
1      1     Rucha    Mahabal     A    Mumbai   CS  80000
3      3    Prachi      Sawant     A    Mumbai   HR  60000
2      2   Anushka  Bommakanty     B    Mumbai   IT  50000
4      4   Tanisha      Ramani     A      Pune   CS  40000
5      5     Mahima  Prajapati     B     Nasik   HR  30000
8      8     Bhakti    Somaiya     B Bangalore   CS  28000
6      6     Anisha        Roy     C Bangalore   IT  25000
7      7      Komal   Tripathi     C      Pune   IT  25000
9      9     Priya   Kulkarni     A    Mumbai   IT     NA
```

Q.10) Merging of data set in r-studio

Note:- Create new data set in excel to merge – bonus.csv

	A	B
1	emp_id	bonus
2	1	7000
3	2	8000
4	3	4000
5	4	3000
6	5	2000
7	6	1000
8	7	2000
9	8	3000
10	9	1000

Output:-

```
> employee2 <- read.csv(file.choose())
> employee2
  emp_id bonus
1      1  7000
2      2  8000
3      3  4000
4      4  3000
5      5  2000
6      6  1000
7      7  2000
8      8  3000
9      9  1000
```

i) Default Merging

Output:-

```
> employee_bonus <- merge(emp, employee2, by="emp_id")
> employee_bonus
  emp_id first_name  last_name grade       loc dept salary bonus
1      1       Rucha    Mahabal     A    Mumbai   CS  80000  7000
2      2    Anushka  Bommakanty     B    Mumbai   IT  50000  8000
3      3     Prachi      Sawant     A    Mumbai   HR  60000  4000
4      4    Tanisha      Ramani     A      Pune   CS  40000  3000
5      5      Mahima  Prajapati     B     Nasik   HR  30000  2000
6      6      Anisha         Roy     C Bangalore   IT  25000  1000
7      7       Komal   Tripathi     C      Pune   IT  25000  2000
8      8      Bhakti    Somaiya     B Bangalore   CS  28000  3000
9      9       Priya   Kulkarni     A    Mumbai   IT     NA  1000
```

ii) left merge

Output:-

```
> left_merge <- merge(emp, employee2, by="emp_id", all.x=TRUE)
> left_merge
  emp_id first_name  last_name grade       loc dept salary bonus
1      1       Rucha    Mahabal     A    Mumbai   CS  80000  7000
2      2    Anushka  Bommakanty     B    Mumbai   IT  50000  8000
3      3     Prachi      Sawant     A    Mumbai   HR  60000  4000
4      4    Tanisha      Ramani     A      Pune   CS  40000  3000
5      5      Mahima  Prajapati     B     Nasik   HR  30000  2000
6      6      Anisha         Roy     C Bangalore   IT  25000  1000
7      7       Komal   Tripathi     C      Pune   IT  25000  2000
8      8      Bhakti    Somaiya     B Bangalore   CS  28000  3000
9      9       Priya   Kulkarni     A    Mumbai   IT     NA  1000
```

ii) right merge

```
> right_merge <- merge(emp, employee2, by="emp_id", all.y=TRUE)
> right_merge
  emp_id first_name  last_name grade       loc dept salary bonus
1      1       Rucha    Mahabal     A    Mumbai   CS  80000  7000
2      2    Anushka  Bommakanty     B    Mumbai   IT  50000  8000
3      3     Prachi      Sawant     A    Mumbai   HR  60000  4000
4      4    Tanisha      Ramani     A      Pune   CS  40000  3000
5      5      Mahima  Prajapati     B     Nasik   HR  30000  2000
6      6      Anisha         Roy     C Bangalore   IT  25000  1000
7      7       Komal   Tripathi     C      Pune   IT  25000  2000
8      8      Bhakti    Somaiya     B Bangalore   CS  28000  3000
9      9       Priya   Kulkarni     A    Mumbai   IT     NA  1000
```

PRACTICAL 3: Practical of Principal Component Analysis

1) Iris Data Set

```
> data("iris")
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
>
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
16           5.7         4.4          1.5         0.4     setosa
17           5.4         3.9          1.3         0.4     setosa
18           5.1         3.5          1.4         0.3     setosa
19           5.7         3.8          1.7         0.3     setosa
20           5.1         3.8          1.5         0.3     setosa
21           5.4         3.4          1.7         0.2     setosa
22           5.1         3.7          1.5         0.4     setosa
23           4.6         3.6          1.0         0.2     setosa
24           5.1         3.3          1.7         0.5     setosa
25           4.8         3.4          1.9         0.2     setosa
26           5.0         3.0          1.6         0.2     setosa
27           5.0         3.4          1.6         0.4     setosa
28           5.2         3.5          1.5         0.2     setosa
29           5.2         3.4          1.4         0.2     setosa
30           4.7         3.2          1.6         0.2     setosa
31           4.8         3.1          1.6         0.2     setosa
32           5.4         3.4          1.5         0.4     setosa
33           5.2         4.1          1.5         0.1     setosa
34           5.5         4.2          1.4         0.2     setosa
35           4.9         3.1          1.5         0.2     setosa
36           5.0         3.2          1.2         0.2     setosa
```

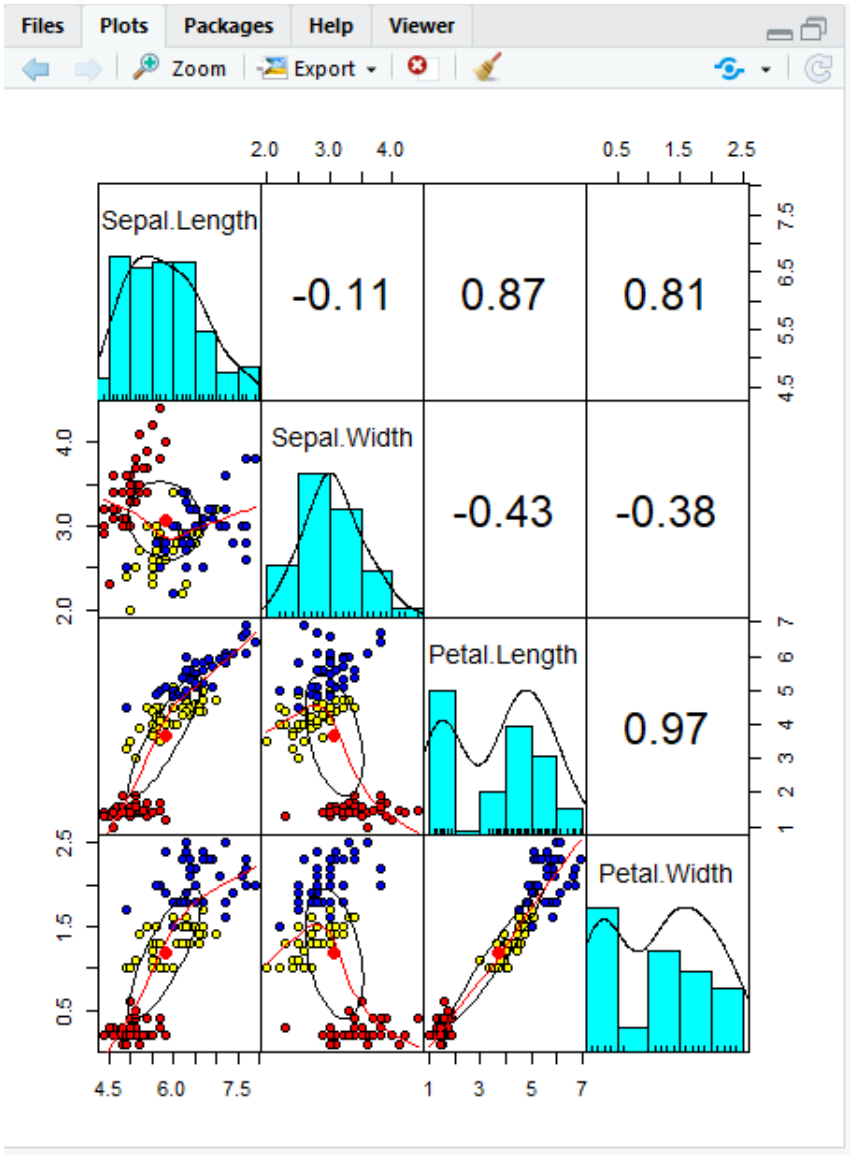
2) partition Data

```
> Ind = sample(2, nrow(iris), replace = TRUE, prob=c(0.8,0.2))
> Training = iris[Ind==1,]
> Testing = iris[Ind==2,]
```

3) plot the data

```
install.packages("psych")
library(psych)
pairs.panels(training[1:4],gap=0,bg=c("yellow","red","blue")[training$
Species],pch = 21)
```

Output:



4) Principal Component Analysis(PCA)

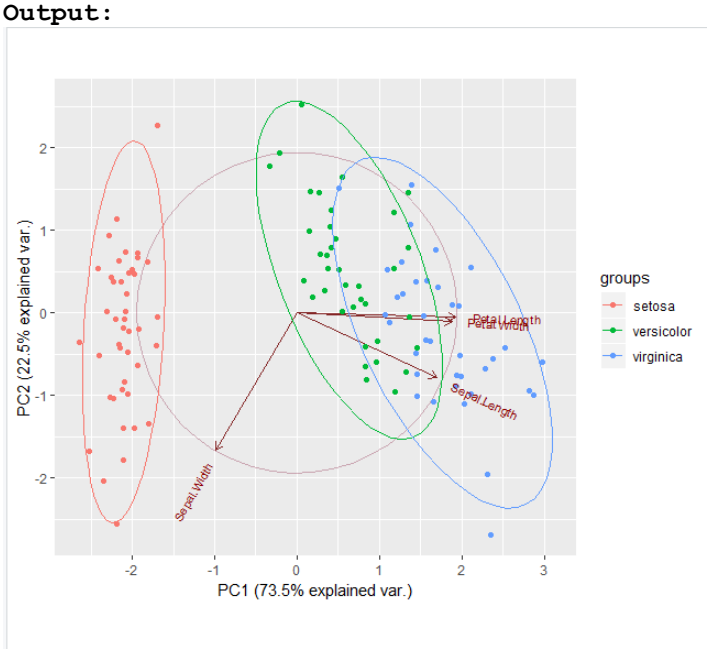
```
> pc = prcomp(Training[,-5],center = TRUE,scale.= TRUE)
> attributes(pc)
$`names`
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"

> pc$scale
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.8492205    0.4632086    1.7851316    0.7789856
> pc
Standard deviations (1, .., p=4):
[1] 1.7091386 0.9566835 0.3819434 0.1331211

Rotation (n x k) = (4 x 4):
                    PC1        PC2        PC3        PC4
Sepal.Length  0.5188970 0.38766308 -0.7156377  0.2613921
Sepal.Width  -0.2720814 0.91952449  0.2570020 -0.1199851
Petal.Length  0.5806727 0.02726548  0.1432447 -0.8009724
Petal.Width   0.5652759 0.05872521  0.6334774  0.5250914
>
```

5) ggbiplot

```
install.packages("devtools")
library(devtools)
install_github("vqv/ggbiplot")
library(ggbiplot)
```

```
> G = ggbiplot(pc, obs.scale=1,
+              var.scale=1,
+              groups=Training$Species,
+              ellipse=TRUE,
+              circle=TRUE,
+              ellipse.prob=0.95)
> G
>
```

Output:



PRACTICAL 4: Practical of Clustering

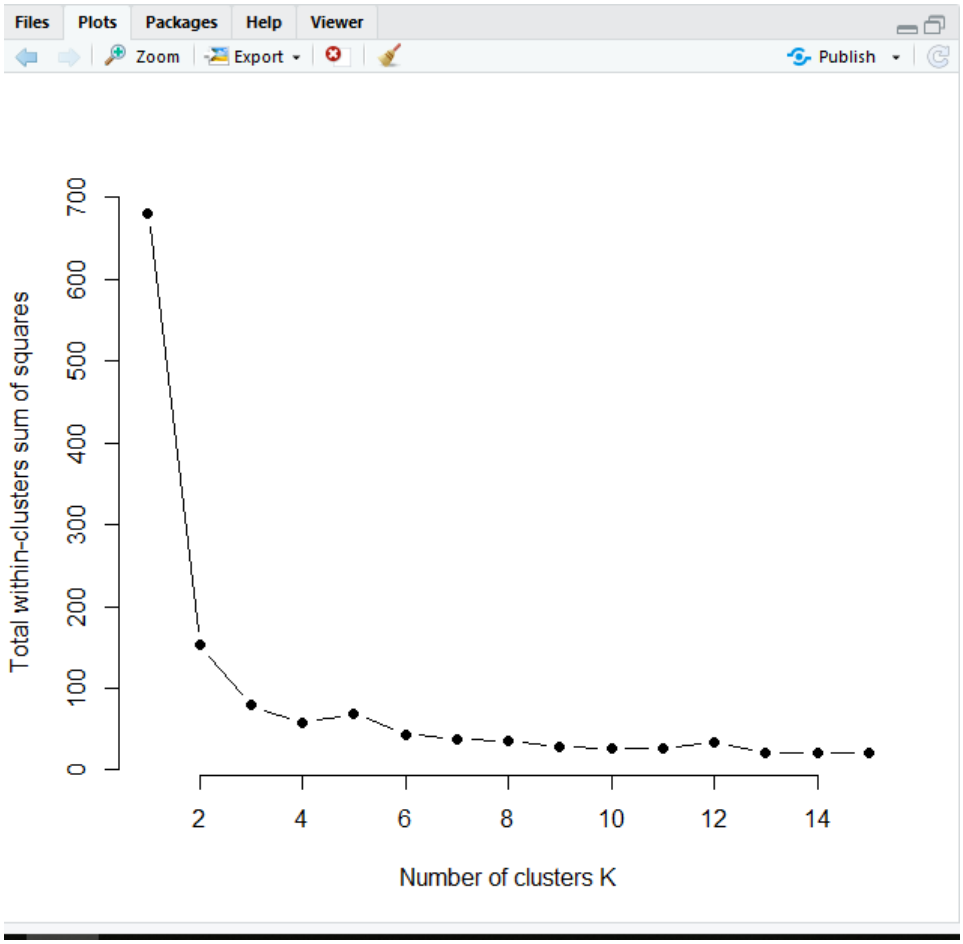
1) k-means clustering

```
> 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
15           5.8         4.0          1.2         0.2
16           5.7         4.4          1.5         0.4
17           5.4         3.9          1.3         0.4
18           5.1         3.5          1.4         0.3
19           5.7         3.8          1.7         0.3
20           5.1         3.8          1.5         0.3
21           5.4         3.4          1.7         0.2
22           5.1         3.7          1.5         0.4
23           4.6         3.6          1.0         0.2
24           5.1         3.3          1.7         0.5
25           4.8         3.4          1.9         0.2
26           5.0         3.0          1.6         0.2
27           5.0         3.4          1.6         0.4
28           5.2         3.5          1.5         0.2
29           5.2         3.4          1.4         0.2
30           4.7         3.2          1.6         0.2
31           4.8         3.1          1.6         0.2
32           5.4         3.4          1.5         0.4
33           5.2         4.1          1.5         0.1
34           5.5         4.2          1.4         0.2
35           4.9         3.1          1.5         0.2
36           5.0         3.2          1.2         0.2
37           5.5         3.5          1.3         0.2
38           4.9         3.6          1.4         0.1
39           4.4         3.0          1.3         0.2
40           5.1         3.4          1.5         0.2
41           5.0         3.5          1.3         0.3
42           4.5         2.3          1.3         0.3
43           4.4         3.2          1.3         0.2
44           5.0         3.5          1.6         0.6
45           5.1         3.8          1.9         0.4
46           4.8         3.0          1.4         0.3
```

```
> 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
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  1   1   1   1   1   1   1   1   1   1   3   3   2   3   3   3   3   3   3   3
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3   3
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  2   3   2   2   2   2   3   2   2   2   2   2   2   3   3   2   2   2   2   3
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  2   3   2   3   2   2   3   3   2   2   2   2   2   3   2   2   2   2   3   2
141 142 143 144 145 146 147 148 149 150
  2   2   3   2   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"
[6] "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  69.24240  43.68323  38.11226
 [8]  35.92851  27.78609  26.76674  26.35376  34.11821  21.55400  21.23139
[15]  20.59309
>
> plot(1:15,wss,type="b",pch=19,frame=FALSE,xlab="Number of clusters K",
+      ylab="Total within-clusters sum of squares")
```

Output of plot:

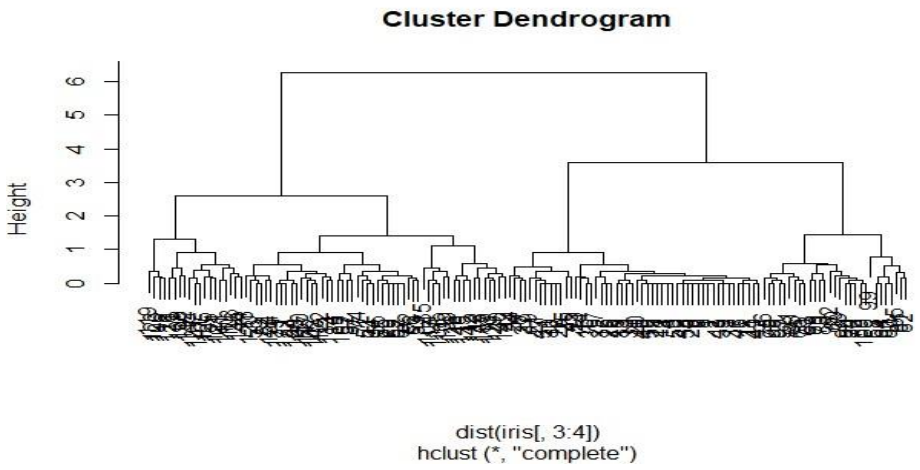


Q.2) Complete agglomerative clustering

i) cluster

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

Output:-



ii) clustercut

Output:-

```
> clustercut<-cutree(clusters,3)
> clustercut
  [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 1 1 1 1 1 1 1
 [43] 1 1 1 1 1 1 1 1 2 2 2 3 2 2 2 3 2 3 3 3 3 2 3 3 2 3 2 3 2 3 2 2 3 3 2 2 2 3 3 3 3 2
 [85] 2 2 2 3 3 3 3 2 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[127] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

iii) table(Create table of specified data set)

table(clustercut,iris$Species)

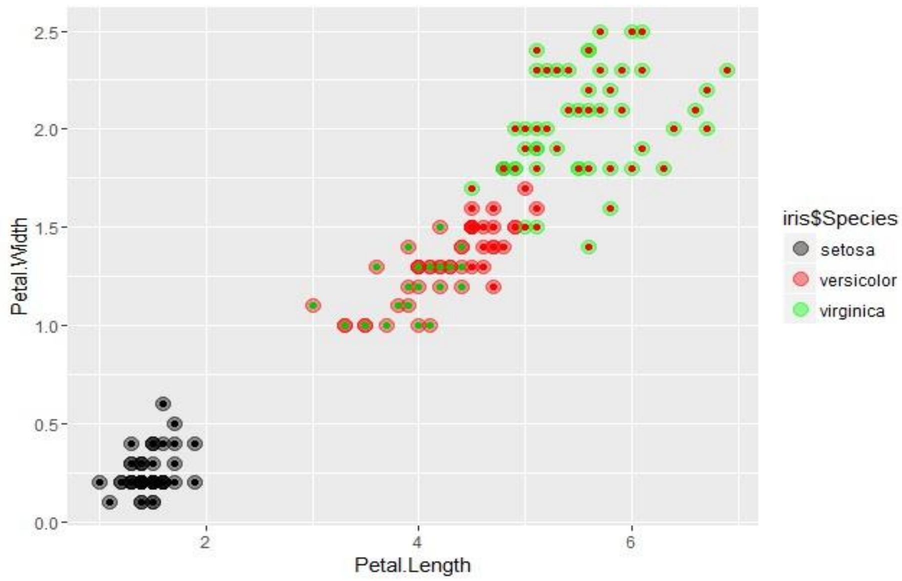
Output:-

```
> table(clustercut,iris$Species)

clustercut setosa versicolor virginica
         1     50          0         0
         2      0         21        50
         3      0         29         0
```

iv)ggplot in average agglomerative cluster

```
library(ggplot2)
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:-



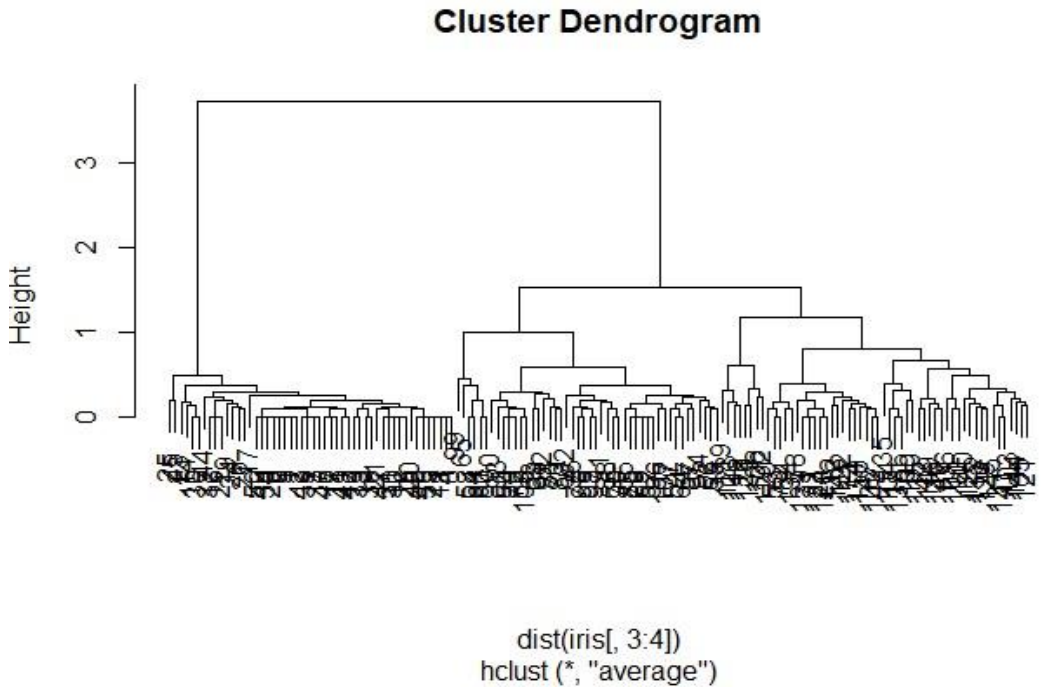
Q.3) Average agglomerative clustering

i) Cluster

Output:-

```
> plot(clusters)
> clusters1<-hclust(dist(iris[,3:4]),method = 'average')
> clusters1

Call:
hclust(d = dist(iris[, 3:4]), method = "average")

Cluster method   : average
Distance         : euclidean
Number of objects: 150
```

Output:



ii) Clustercut

Output:-

```
> clustercut<-cutree(clusters,3)
> clustercut
  [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 1 1 1 1 1 1 1 1 1 1 1 1 1
 [43] 1 1 1 1 1 1 1 1 2 2 2 3 2 2 2 3 2 3 3 3 3 2 3 3 2 3 2 3 2 3 2 2 3 3 2 2 2 3 3 3 3 2
 [85] 2 2 2 3 3 3 3 2 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[127] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

iii) table(Create table of specify data set)

```
table(clustercut,iris$Species)
```

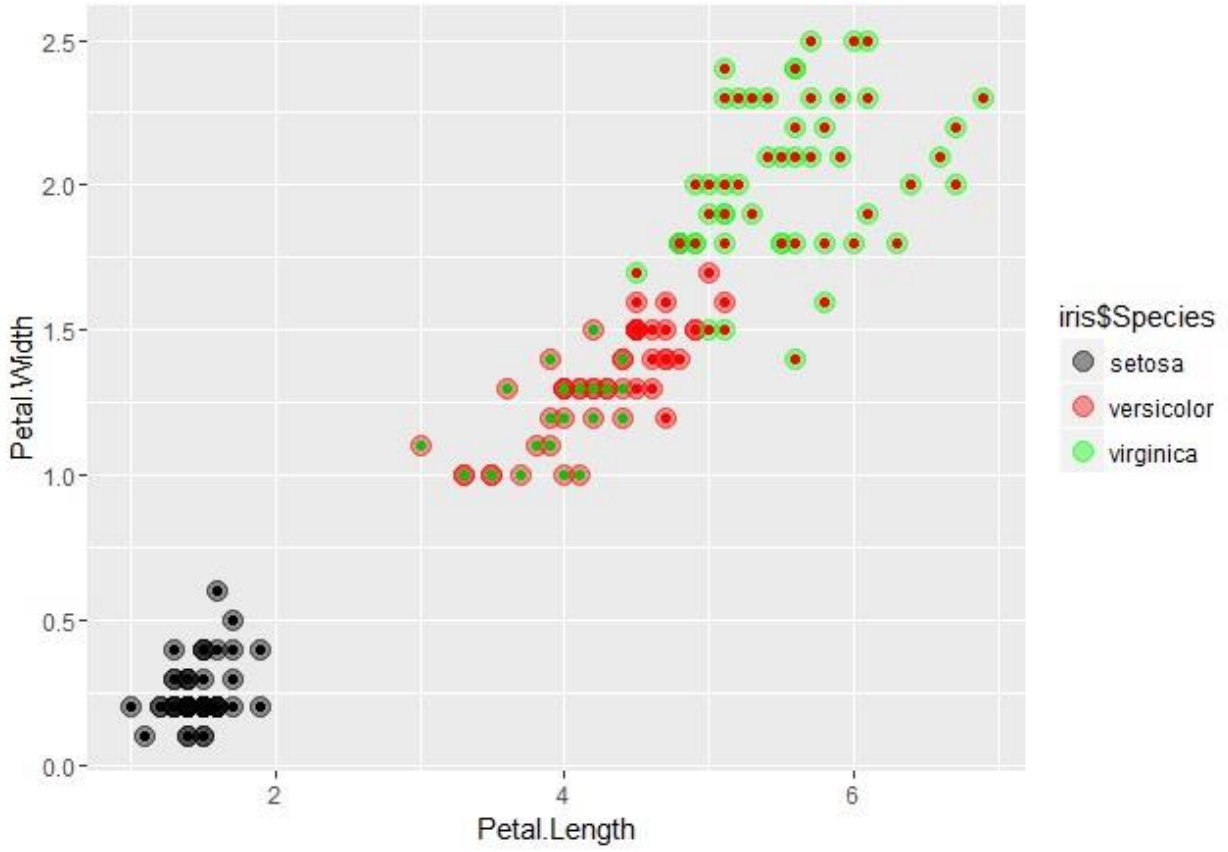
Output:-

```
> table(clustercut,iris$Species)

clustercut setosa versicolor virginica
         1     50          0         0
         2      0         21        50
         3      0         29         0
```

iv)ggplot in average agglomerative cluster

```
 library(ggplot2)
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:-



PRACTICAL 5: Practical of Time-Series Forecasting

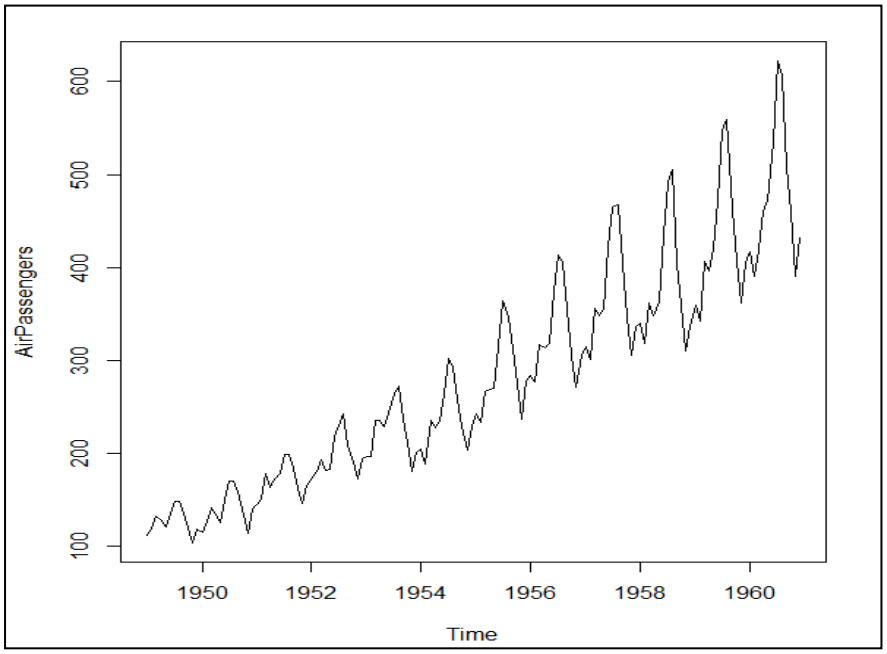
Aim: Practical of time series

Step 1: Adding Data Air Passenger and information about data

```
data(AirPassengers)
class(AirPassengers)
start(AirPassengers)
end(AirPassengers)
frequency(AirPassengers)
summary(AirPassengers)
```

```
> 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)
```



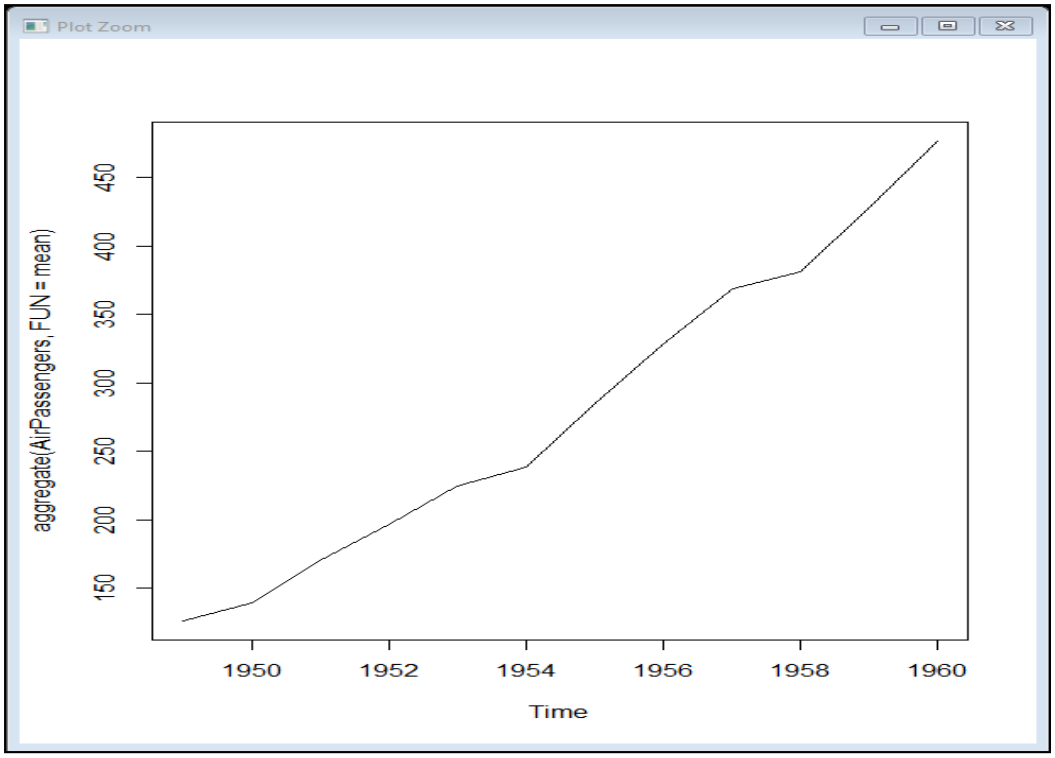
Step 2:

```
abline(reg=lm(AirPassengers~time(AirPassengers)))
cycle(AirPassengers)
```

```
> abline(reg=lm(AirPassengers~time(AirPassengers)))
> 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
```

Step 3:

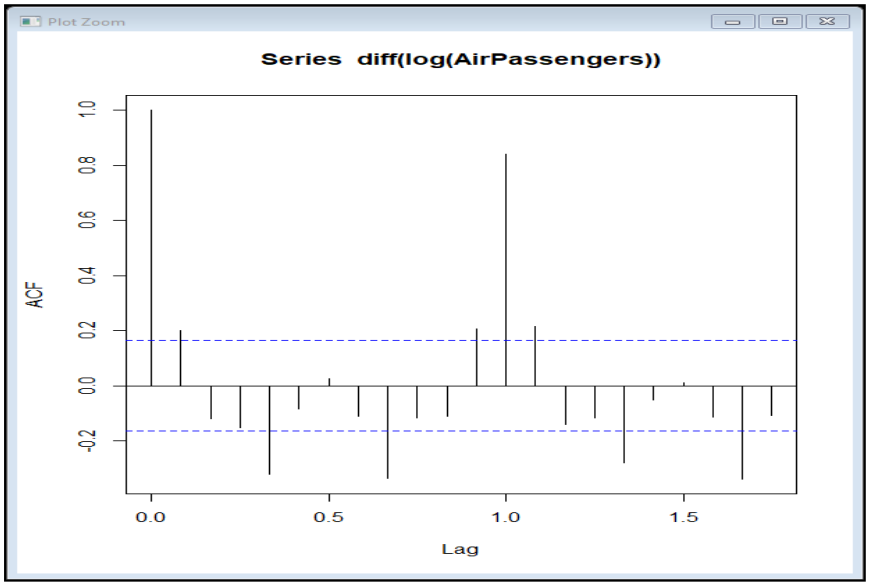
```
plot(aggregate(AirPassengers,FUN=mean))
```



Step 4:

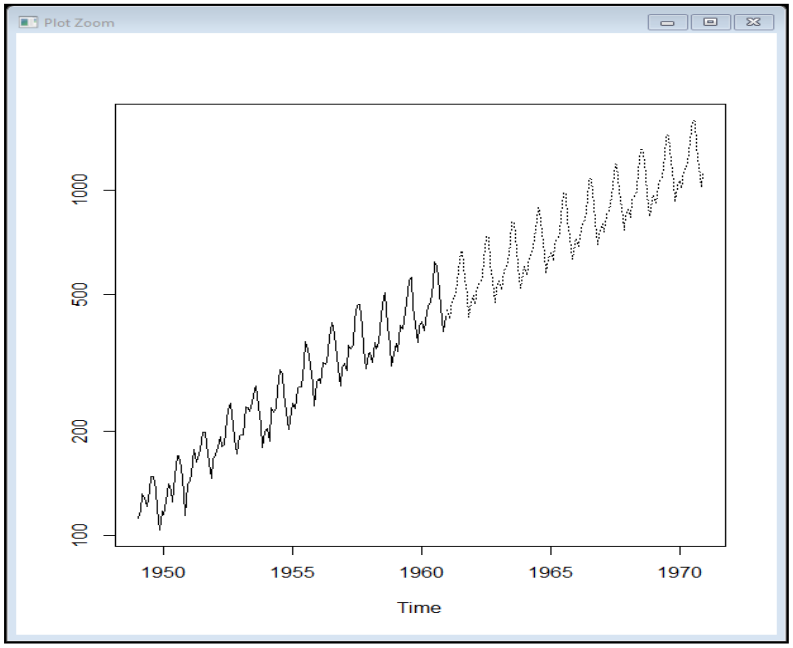
```
boxplot(AirPassengers~cycle(AirPassengers))
acf(log(AirPassengers))
acf(diff(log(AirPassengers)))
(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1),
period = 12)))
```

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

Call:
arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0,
    1, 1), period = 12))

Coefficients:
          ma1     sma1
      -0.4018  -0.5569
s.e.   0.0896   0.0731

sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -483.4
```



Step 5 :

```
pred <- predict(fit, n.ahead = 10*12)
ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))
```



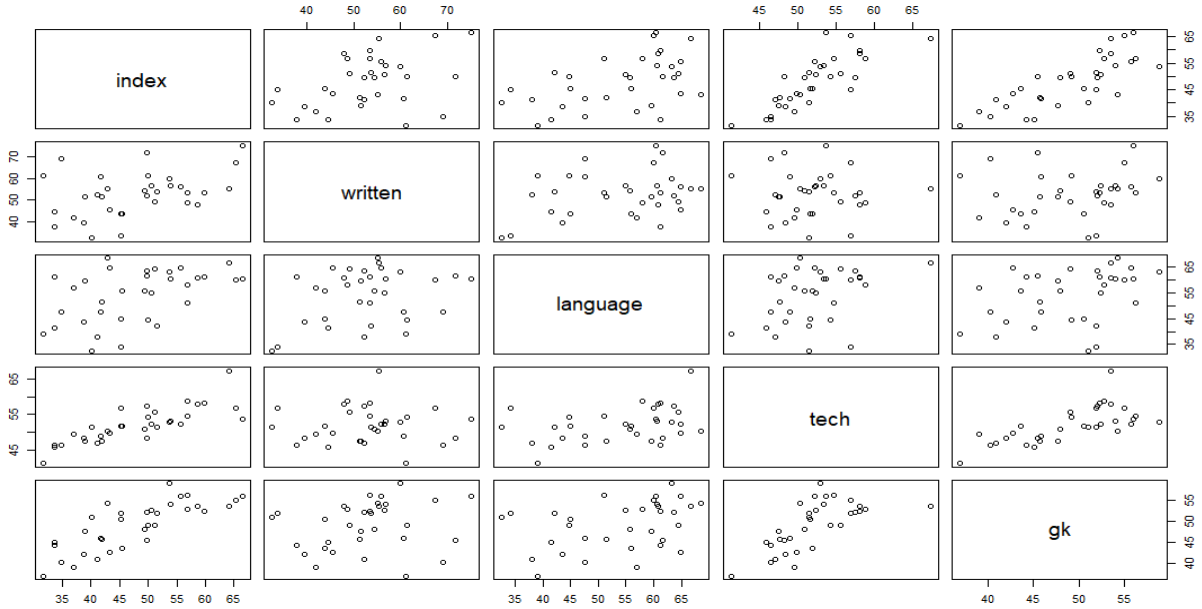
PRACTICAL 6: Practical of Simple/Multiple Linear Regression

Aim: Practical of Simple and Multiple Regression

Step 1:- Import file index.csv

Scatter plot

```
[Workspace loaded from ~/.RData]

> index<-read.csv(file.choose(),sep = ",",header = T)
> names(index)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
> pairs(~index+written+language+tech+gk,data=index)
>
```



Step 2:- Fit a model (elements are stored)

```
> summary(model1)

Call:
lm(formula = index ~ ., data = index)

Residuals:
    Min      1Q  Median      3Q     Max
-5.5382 -2.4528  0.0266  2.2774  5.4622

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -56.37329    7.12537  -7.912 1.67e-08 ***
empid        -0.12830    0.06542  -1.961  0.06025 .
written       0.33206    0.06472   5.131 2.14e-05 ***
language      0.04794    0.06828   0.702  0.48859
tech          1.17174    0.17714   6.615 4.26e-07 ***
gk            0.51787    0.15123   3.424  0.00198 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.382 on 27 degrees of freedom
Multiple R-squared:  0.8922,    Adjusted R-squared:  0.8722
F-statistic: 44.67 on 5 and 27 DF,  p-value: 3.188e-12
```

Index= -56.3724+(-
0.12830)+written(0.33206)+language(0.04794)+tech(1.17174)+gk(0.57787)

Conclusion:-

As value of multiple r(square) is 0.8922

So 89% of variation in index is explained by the model and 11% is not explained by the model

Step 3 :- Write down the equation and check for global testing

```
> index$pred<-fitted(model1)
> head(index)
  empid index written language  tech    gk     pred
1     1 45.52   43.83    55.92 51.82 43.58 44.02220
2     2 40.10   32.71    32.56 51.49 51.03 42.55279
3     3 50.61   56.64    54.84 52.29 52.47 53.12213
4     4 38.97   51.53    59.69 47.48 47.69 43.41803
5     5 41.87   51.35    51.50 47.59 45.77 41.97188
6     6 38.71   39.60    43.63 48.34 42.06 36.52202
> index$res<-residuals(model1)
> head(index)
  empid index written language  tech    gk     pred        res
1     1 45.52   43.83    55.92 51.82 43.58 44.02220  1.4978042
2     2 40.10   32.71    32.56 51.49 51.03 42.55279 -2.4527900
3     3 50.61   56.64    54.84 52.29 52.47 53.12213 -2.5121304
4     4 38.97   51.53    59.69 47.48 47.69 43.41803 -4.4480298
5     5 41.87   51.35    51.50 47.59 45.77 41.97188 -0.1018831
6     6 38.71   39.60    43.63 48.34 42.06 36.52202  2.1879775
> |
```

Step 4:- To check the multicollinearity.

```
> library(car)
> vif(model1)
   empid  written language     tech       gk
1.119815 1.185225 1.344122 2.178955 2.033284
> |
```

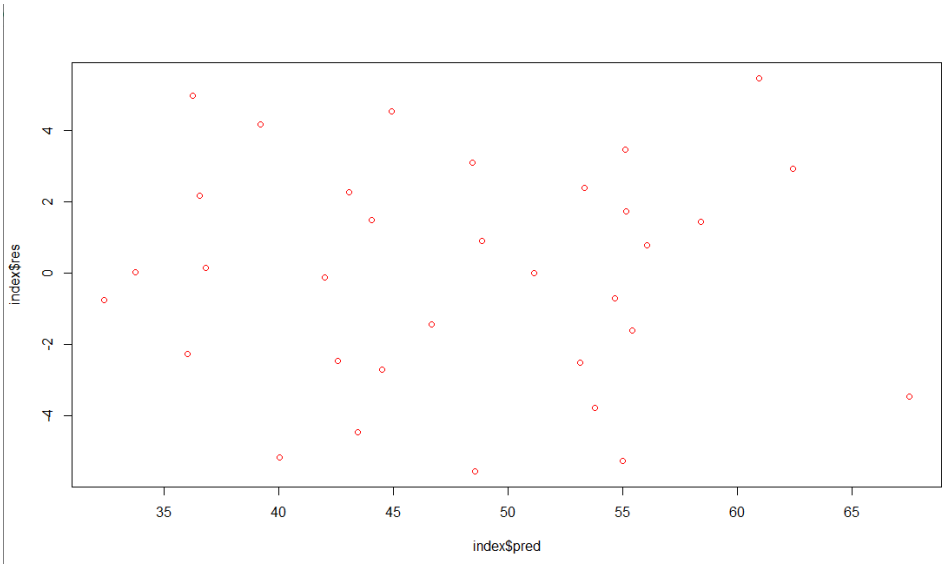
Conclusion:

As all VIF are less than 5

Multicollinearity is not present.

Step 5:- to check heteroscedasticity

```
> plot(index$pred,index$res,col="red")
```



Conclusion:-

Sience errors are generated randomly. There is no heteroscedasticity.

Step 6 :- Perform Shapiro test to check normality of errors.

```
> shapiro.test(index$res)

        Shapiro-Wilk normality test

data:  index$res
W = 0.97172, p-value = 0.5293
```

Conclusion:-

As p value is greater than 0.05 accept Ho.

Step 7 :- Detecting heteroscedasticity of data using NCV test.

```
> library(car)
> ncvTest(model1,~written+language+tech+gk)
Non-constant Variance Score Test
Variance formula: ~ written + language + tech + gk
Chisquare = 2.146914, Df = 4, p = 0.70876
```

Conclusion:-

Ho: There is homoscedasticity

H1: there is no constant variance

As P-value is greater than 0.05 we accept Ho.

Step 8 :- Detecting autocorrelation using Durbin Watson Test d=2(1-r)

```
> library(car)
> durbinWatsonTest(model1)
 lag Autocorrelation D-W Statistic p-value
   1       0.2268414      1.500571   0.084
 Alternative hypothesis: rho != 0
>
```

Conclusion:-

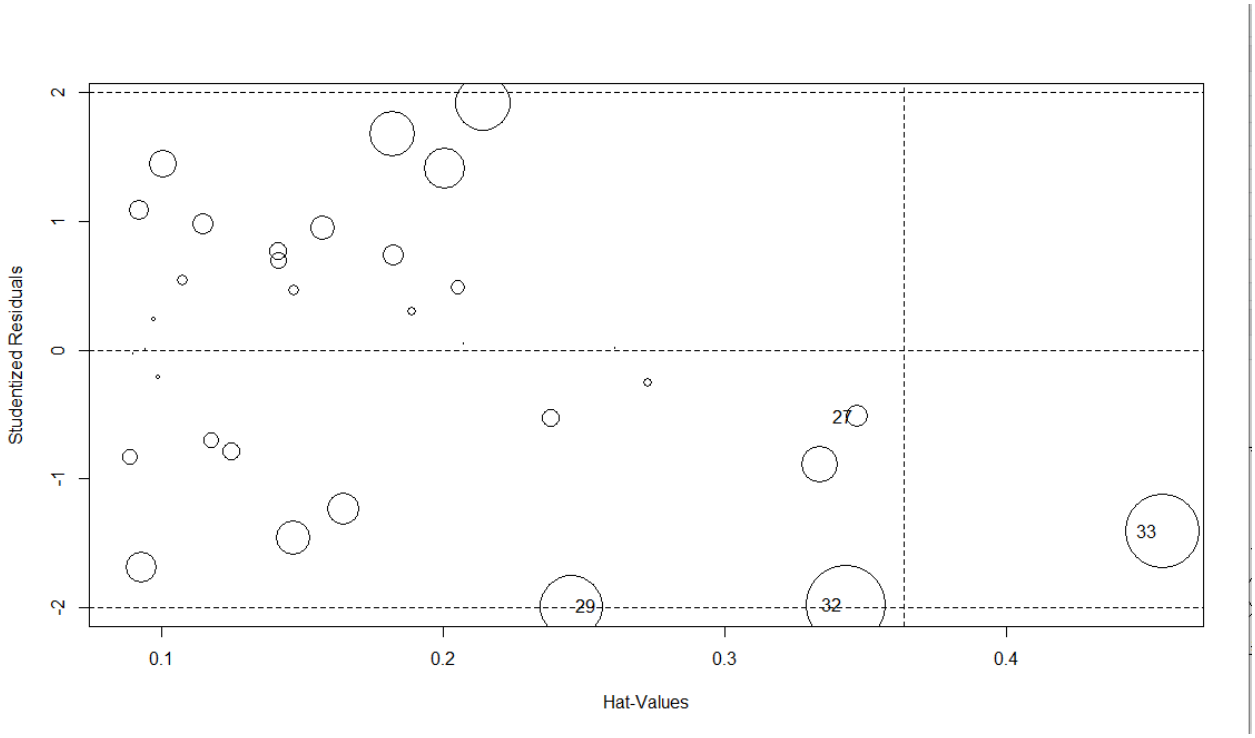
Ho : Auto correlation is not present among errors

H1 : not Ho

As P-value is greater than 0.05 we accept Ho.

Step 9 :- influence plot

```
> influencePlot(model1)
      StudRes       Hat      CookD
27 -0.5173889 0.3473188 0.02440349
29 -1.9854091 0.2453822 0.19264149
32 -1.9789285 0.3433424 0.30800324
33 -1.4060295 0.4554469 0.26594943
```



Step 10 :- Validation using Hold-Out method

```
> library("lattice")
> library("ggplot2")
> library("caret")
> index<-read.csv(file.choose(),sep = ",",header=T)
> summary(index)
     empid          index          written         language          tech
 Min.   : 1   Min.   :31.64   Min.   :32.71   Min.   :32.56   Min.   :41.25
 1st Qu.: 9   1st Qu.:41.19   1st Qu.:45.59   1st Qu.:44.89   1st Qu.:48.34
 Median :17   Median :49.45   Median :53.38   Median :57.04   Median :51.64
 Mean   :17   Mean   :47.87   Mean   :52.66   Mean   :53.99   Mean   :52.02
 3rd Qu.:25   3rd Qu.:53.92   3rd Qu.:56.75   3rd Qu.:61.28   3rd Qu.:54.68
 Max.   :33   Max.   :66.39   Max.   :75.03   Max.   :68.53   Max.   :67.27
       gk
 Min.   :37.00
 1st Qu.:45.07
 Median :50.53
 Mean   :49.04
 3rd Qu.:53.50
 Max.   :58.90
> data<-createDataPartition(index$empid,p=0.8,list = F)
> head(data)
     Resample1
[1,]         1
[2,]         2
[3,]         4
[4,]         5
[5,]         6
[6,]         7
> dim(data)
[1] 29  1
>
```

Step 11 :-

```
> traindata<-index[data,]
> testdata<-index[-data,]
> dim(traindata)
[1] 29  6
> dim(testdata)
[1] 4 6
>
```

Step 12 :- Validation using k fold method

```
> kfolds<-trainControl(method = "cv" , number = 4)
> modelkfold<-train(index~written+language+tech+gk,data = index,method="lm",trControl=kfolds)
> modelkfold
Linear Regression

33 samples
 4 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 24, 25, 25, 25
Resampling results:

  RMSE      Rsquared   MAE
  4.237887  0.8787614  3.595394

Tuning parameter 'intercept' was held constant at a value of TRUE
>
```

Conclusion :-

As the value of RMSE is sufficiently large the model is stable.

Step 13 :- Model selection forward method.

```
> index<-read.csv(file.choose(),sep = ",",header = T)
> null<-lm(index~1,data = index)
> full<-lm(index~.,data = index)
> names(index)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
> step(null,scope = list(lower=null,upper=full),direction = "forward")
Start:  AIC=149.28
index ~ 1

           Df Sum of Sq     RSS    AIC
+ tech      1   1867.81  994.92 116.40
+ gk        1   1787.03 1075.69 118.98
+ language  1    660.54 2202.19 142.62
+ written   1    479.64 2383.09 145.23
<none>                  2862.73 149.28
+ empid     1     62.42 2800.31 150.55

Step:  AIC=116.4
index ~ tech

           Df Sum of Sq    RSS     AIC
+ written   1    490.24 504.68  96.005
+ gk        1    302.78 692.14 106.428
+ language  1     99.24 895.68 114.936
<none>                  994.92 116.403
+ empid     1     24.53 970.39 117.579

Step:  AIC=96
index ~ tech + written
```

```
           Df Sum of Sq     RSS     AIC
+ gk        1   149.196 355.48 86.440
+ empid     1    49.957 454.72 94.565
<none>                  504.68 96.005
+ language  1     7.276 497.40 97.526

Step:  AIC=86.44
index ~ tech + written + gk

           Df Sum of Sq     RSS     AIC
+ empid     1    41.105 314.38 84.385
<none>                  355.48 86.440
+ language  1     2.764 352.72 88.183

Step:  AIC=84.39
index ~ tech + written + gk + empid

           Df Sum of Sq     RSS     AIC
<none>                  314.38 84.385
+ language  1    5.6376 308.74 85.788

Call:
lm(formula = index ~ tech + written + gk + empid, data = index)

Coefficients:
(Intercept)         tech      written           gk        empid
   -56.4681       1.1988       0.3456       0.5276      -0.1233
```

Step 14:- Model selection backword method.

```
> index<-read.csv(file.choose(),sep = ",",header = T)
> null<-lm(index~1,data = index)
> full<-lm(index~.,data = index)
> names(index)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
> step(full,scope = list(lower=null,upper=full),direction = "backward")
Start:  AIC=85.79
index ~ empid + written + language + tech + gk

           Df Sum of Sq    RSS     AIC
- language  1      5.64 314.38  84.385
<none>                308.74  85.788
- empid     1     43.98 352.72  88.183
- gk        1    134.09 442.83  95.691
- written   1    300.99 609.74 106.245
- tech      1    500.35 809.10 115.581

Step:  AIC=84.39
index ~ empid + written + tech + gk

          Df Sum of Sq    RSS     AIC
<none>                314.38  84.385
- empid    1     41.11 355.48  86.440
- gk       1    140.34 454.72  94.565
- written  1    357.94 672.32 107.469
- tech     1    549.77 864.15 115.753

Call:
lm(formula = index ~ empid + written + tech + gk, data = index)

Coefficients:
(Intercept)        empid      written         tech           gk
   -56.4681      -0.1233       0.3456       1.1988       0.5276

> |
```

PRACTICAL 7: Practical of Logistic Regression

1) There is inbuilt data present in R_Studio that is "iris" iris

Output:-

```
> 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
```

```
Summary(iris)
```

Output:-

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

2) Structure of data set

```
str(iris)
```

Output:-

```
> str(ir_data)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

3) levels give you attribute of variable means data set

```
ir_data<-iris
 levels(ir_data$Species)
```

Output:-

```
> levels(ir_data$Species)
[1] "setosa"     "versicolor" "virginica"
```

4) Sample will give you sample data from the data set randomly

Output:-

```
> samp<-sample(1:100,80)
> samp
 [1] 31 26 55  6 45 46 77 35 51 16 57 79 25 93 66 90 18 30 83 56 43 85 42 58 32 13 94 65
[29] 40 20 87 73 24 64 95 74 12 72 62  8 71 52 86 48 34 28 80 47 11 91 17 10 67 75 98 89
[57] 97 49 99  9 19 78 37 39 50 53 92 15 61 22 70 54 36 27 84 81 21 38 63  2
```

```
ir_tes<-ir_data[samp,]
ir_tes
```

Output:-

```
> ir_tes<-ir_data[samp,]
> ir_tes
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
31          4.8         3.1          1.6         0.2     setosa
26          5.0         3.0          1.6         0.2     setosa
55          6.5         2.8          4.6         1.5 versicolor
6           5.4         3.9          1.7         0.4     setosa
45          5.1         3.8          1.9         0.4     setosa
```

```
ir_ctrl<-ir_data[-samp,]
Ir_ctrl
```

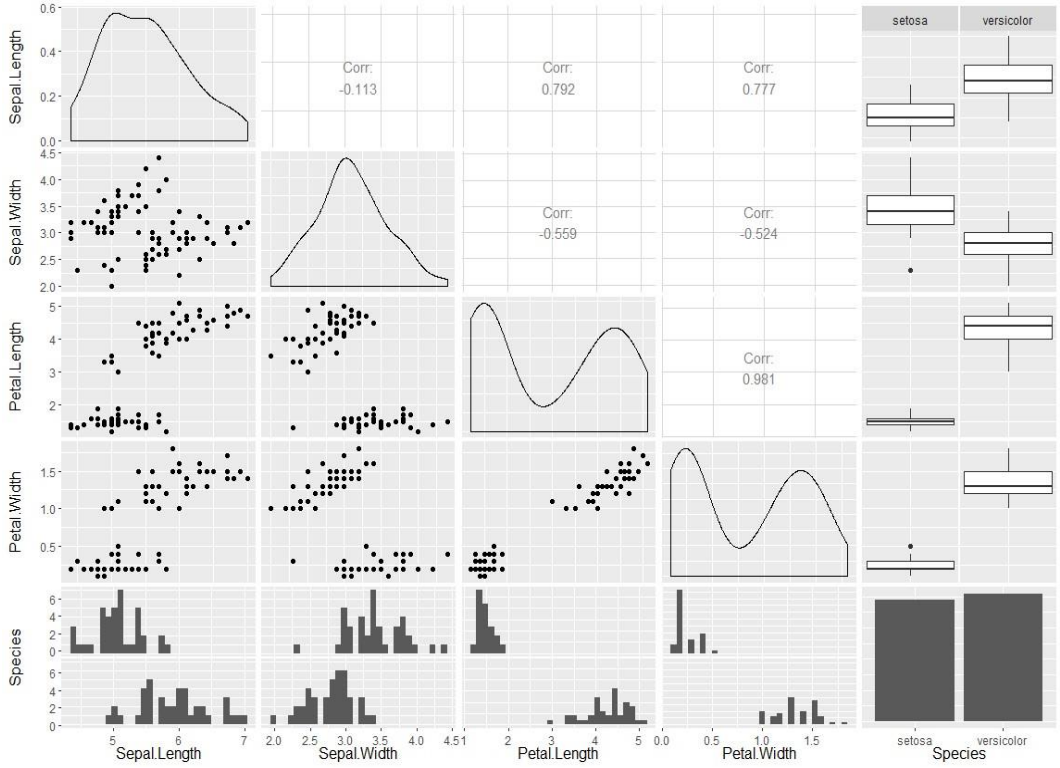
Output:-

```
> ir_ctrl<-ir_data[-samp,]
> ir_tes
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
31          4.8         3.1          1.6         0.2     setosa
26          5.0         3.0          1.6         0.2     setosa
55          6.5         2.8          4.6         1.5 versicolor
6           5.4         3.9          1.7         0.4     setosa
45          5.1         3.8          1.9         0.4     setosa
46          4.8         3.0          1.4         0.3     setosa
77          6.8         2.8          4.8         1.4 versicolor
```

5)ggplot

```
install.packages("ggplot2")
library(ggplot2)
install.packages("GGally")
library(GGally)
ggpairs(ir_tes)
```

Output:-



PRACTICAL 8: Practical of Hypothesis Testing

Data set

	A	B	C	D	E
		BS	Fasting	pp	
1		194	90	120	
2		90	80	126	
3		96.65	93	140	
4		56.65	97	160	
5		69.56	100	130	
6		100	89	140	
7		96.6	96	150	
8		100.65	89	120	
9		150.23	93	145	
10		56.6	100	135	

Q.1)Create data value object

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

Output:-

```
       BS Fasting  pp
1  194.00      90 120
2   90.00      80 126
3   96.65      93 140
4   56.65      97 160
5   69.56     100 130
6  100.00      89 140
7   96.60      96 150
8  100.65      89 120
9  150.23      93 145
10  56.60     100 135
```

Q.2)Performing t-test

```
t.test(data1$Fasting,data1$Fasting,alternative = "greater",paired=T)
```

Output:-

```
        Paired t-test

data:  data1$Fasting and data1$Fasting
t = NaN, df = 9, p-value = NA
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 NaN NaN
sample estimates:
mean of the differences
                      0
```

Q.3)T-test for variance

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

```
var.test(var$Fasting,var$pp,alternative = "two.sided")
```

Output:-

```
        F test to compare two variances

data:  var$Fasting and var$pp
F = 0.217, num df = 9, denom df = 9, p-value = 0.0327
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.05390025 0.87364915
sample estimates:
ratio of variances
         0.2170021
```

Q.4)T-test for Correlation

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

cor.test(corr$Apti.score,corr$job_pro,alternative =
"two.sided",method= "pearson")
```

Output:-

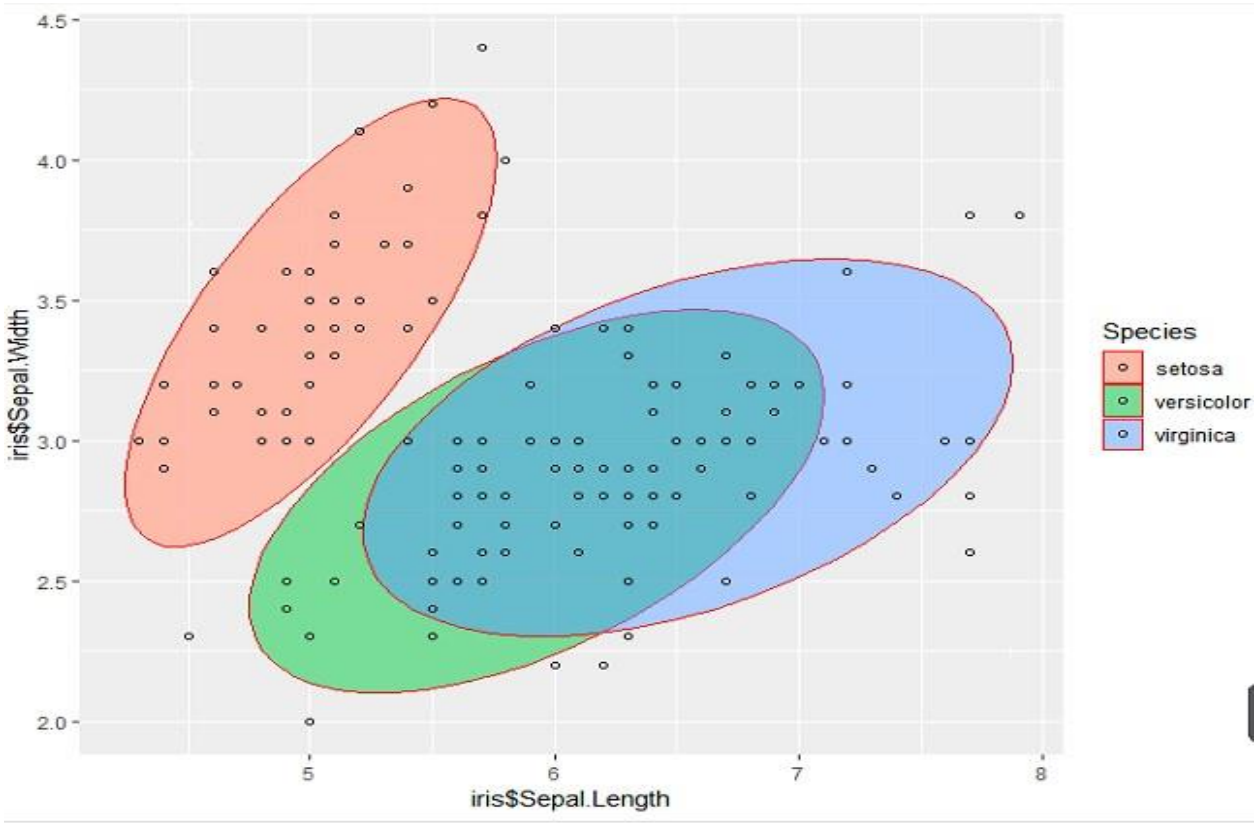
```
> summary(corr)
   Apti.score       job_pro          X            X.1          X.2
 Min.   : 5.00   Min.   :  5.00   Mode:logical      :19   Min.   :32.05
 1st Qu.:29.75   1st Qu.: 28.00   NA's:20        S.D: 1   1st Qu.:32.05
 Median :65.00   Median : 58.50                          Median :32.05
 Mean   :54.60   Mean   : 53.65                          Mean   :32.05
 3rd Qu.:77.75   3rd Qu.: 79.75                          3rd Qu.:32.05
 Max.   :99.00   Max.   :100.00                          Max.   :32.05
                                                         NA's   :19
> cor.test(corr$Apti.score,corr$job_pro,alternative = "two.sided",method = "pearson")

        Pearson's product-moment correlation

data:  corr$Apti.score and corr$job_pro
t = 8.7599, df = 18, p-value = 6.574e-08
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7602711 0.9601307
sample estimates:
      cor
0.8999998
```

- **Correlation**

Note: For ggplot2 Rstudio version must be 3.5.2

```
i)correlation
install packages("ggplot2")
library(ggplot2
ggplot(iris,aes(iris$Sepal.Length,iris$Sepal.Width,col=Species,fill=Sp
ecies))+stat_ellipse(geom="polygon",col="red",alpha=0.5)+geom_point(sh
ape=1,col="black")
```

Output:-


4.5
4.0
3.5
3.0
2.5
2.0
iris$Sepal.Width
5
6
7
8
iris$Sepal.Length
Species
setosa
versicolor
virginica

PRACTICAL 9: Practical of Analysis of Variance

```
> y1 = c(18.2, 20.1, 17.7, 16.8, 18.8, 19.7, 19.1)
> y2 = c(17.4, 18.7, 19.1, 16.4, 15.9, 18.4, 17.7)
> y3 = c(15.2, 18.8, 17.7, 16.5, 15.9, 17.1, 16.7)
> y = c(y1,y2,y3)
> help(rep)
> n = rep(7,3)
> n
[1] 7 7 7
> group = rep(1:3,n)
> group
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3
> help(tapply)
> tmp = tapply(y, group,stem)

  The decimal point is at the |

  16 | 8
  17 | 7
  18 | 28
  19 | 17
  20 | 1


  The decimal point is at the |

  15 | 9
  16 | 4
  17 | 47
  18 | 47
  19 | 1


  The decimal point is at the |
```

s

```
  15 | 29
  16 | 57
  17 | 17
  18 | 8

> tmpfn = function(x)c(sum=sum(x), mean = mean(x), var=var(x), n=length(x))
> tapply(y, group, tmpfn)
$`1`
       sum       mean        var          n
130.400000  18.628571   1.325714   7.000000

$`2`
       sum       mean        var          n
123.600000  17.657143   1.409524   7.000000

$`3`
       sum       mean        var          n
117.900000  16.842857   1.392857   7.000000
```

```
> data = data.frame(y=y, group=factor(group))
> fit = lm(y~ group,data)
> anova(fit)
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
group      2 11.190  5.5948  4.0659 0.03491 *
Residuals 18 24.769  1.3760
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> df = anova(fit)[,"Df"]
> names(df) = c("trt", "err")
> df
trt err
  2  18
> alpha = c(0.05, 0.01)
> qf(alpha, df['trt'], df['err'], lower.tail = FALSE)
[1] 3.554557 6.012905
> anova(fit)["Residuals", "Sum Sq"]
[1] 24.76857
> anova(fit)["Residuals", "Sum Sq"]/qchisq(c(0.025,0.975), 18, lower.tail = FALSE)
[1] 0.7856459 3.0092741
```

PRACTICAL 10: Practical of Decision Tree

1). Load data set in r_studio, Summary of data set, Display name of column

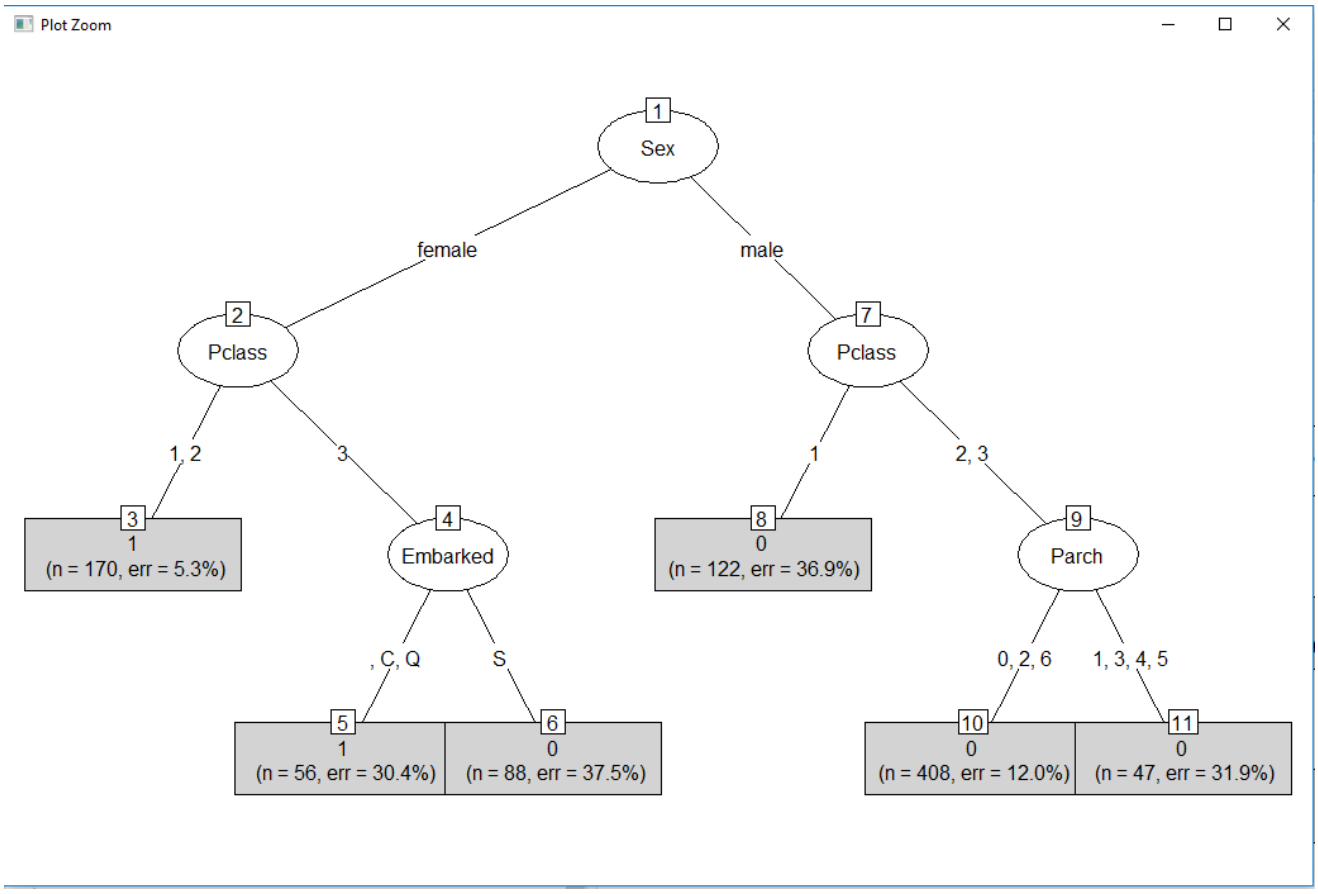
```
> titanic<-read.csv(file.choose(), header = T, sep=",")
> summary(titanic)
  PassengerId       Survived          Pclass
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000
 1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000
 Median :446.0   Median :0.0000   Median :3.000
 Mean   :446.0   Mean   :0.3838   Mean   :2.309
 3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
 Max.   :891.0   Max.   :1.0000   Max.   :3.000

                                     Name         Sex           Age
 Abbing, Mr. Anthony                   :  1   female:314   Min.   : 0.42
 Abbott, Mr. Rossmore Edward           :  1   male  :577   1st Qu.:20.12
 Abbott, Mrs. Stanton (Rosa Hunt)      :  1                Median :28.00
 Abelson, Mr. Samuel                   :  1                Mean   :29.70
 Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
 Adahl, Mr. Mauritz Nils Martin        :  1                Max.   :80.00
 (Other)                               :885                NA's   :177
     SibSp           Parch             Ticket         Fare
 Min.   :0.000   Min.   :0.0000   1601     :  7   Min.   :  0.00
 1st Qu.:0.000   1st Qu.:0.0000   347082   :  7   1st Qu.:  7.91
 Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
 Mean   :0.523   Mean   :0.3816   3101295  :  6   Mean   : 32.20
 3rd Qu.:1.000   3rd Qu.:0.0000   347088   :  6   3rd Qu.: 31.00
 Max.   :8.000   Max.   :6.0000   CA 2144  :  6   Max.   :512.33
                                  (Other)  :852
         Cabin     Embarked
            :687     :  2
 B96 B98    :  4   C:168
 C23 C25 C27:  4   Q: 77
 G6         :  4   S:644
 C22 C26    :  3
 D          :  3
 (Other)    :186
> names(titanic)
 [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
 [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
[11] "Cabin"       "Embarked"
```

2) Display survived data from data set

install.packages("partykit")
install.packages("CHAID",repos="http://R-Forge.R-project.org",type="source")
library(CHAID) #Chi-Square automatic interaction detection
library(partykit) #Tool used for recursive partitioning

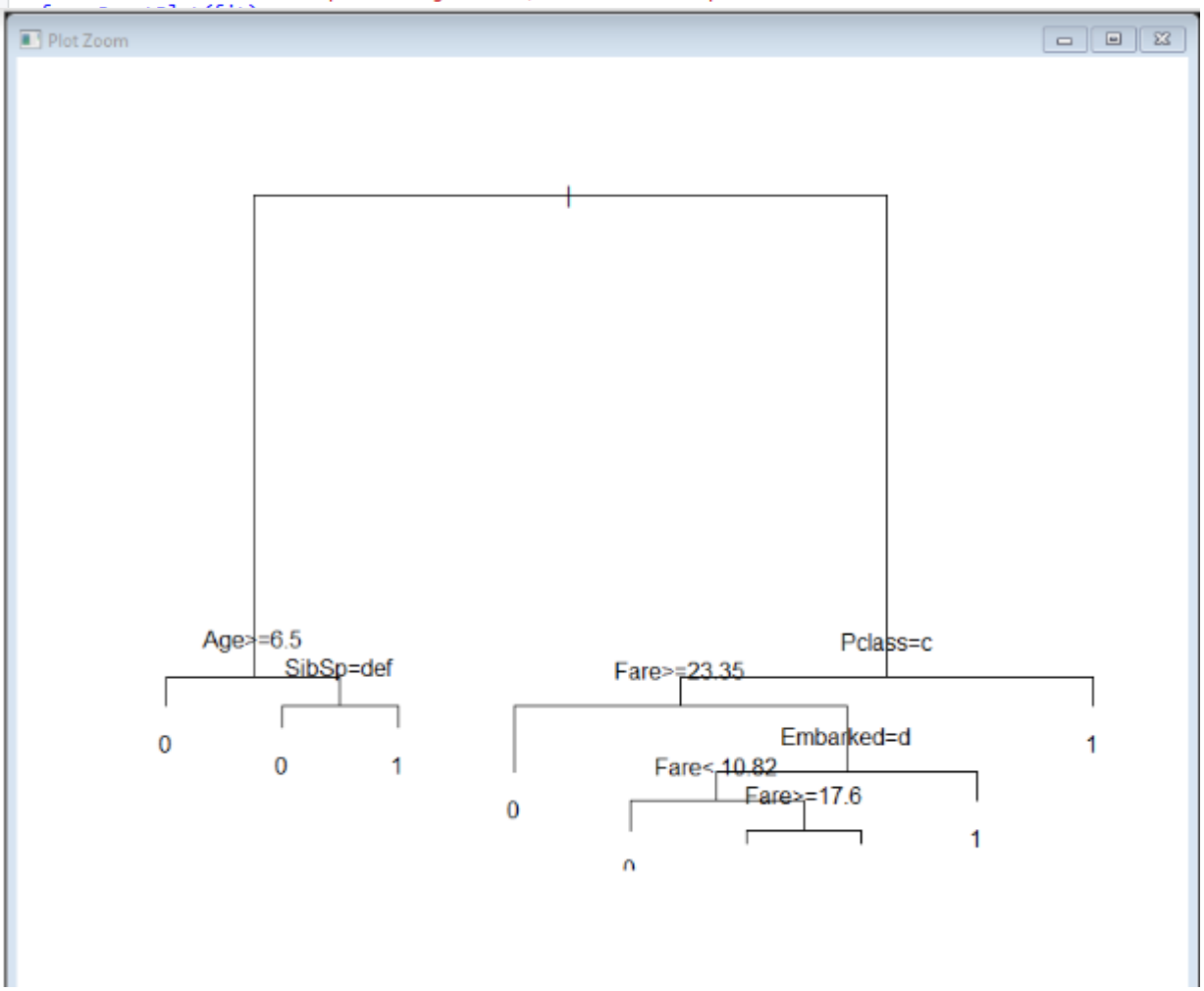
```
> titanic$Survived<-as.factor(titanic$Survived)
> titanic$SibSp<-as.factor(titanic$SibSp)
> titanic$Parch<-as.factor(titanic$Parch)
> titanic$Pclass<-as.factor(titanic$Pclass)
> titanic$Sex<-as.factor(titanic$Sex)
> titanic$Fare<-as.factor(titanic$Fare)
> titanic$Embarked<-as.factor(titanic$Embarked)
> summary(titanic$Survived)
  0   1
549 342
> names(titanic)
 [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
 [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
[11] "Cabin"       "Embarked"
> tree<-chaid(formula=Survived~Pclass+Sex+SibSp+Parch+Fare+Embarked,data=titanic)

> tree<-chaid(formula=Survived~Pclass+Sex+SibSp+Parch+Fare+Embarked,data=titanic)
> class(titanic$Survived)
[1] "factor"
> plot(tree, type="simple")
```

Output:

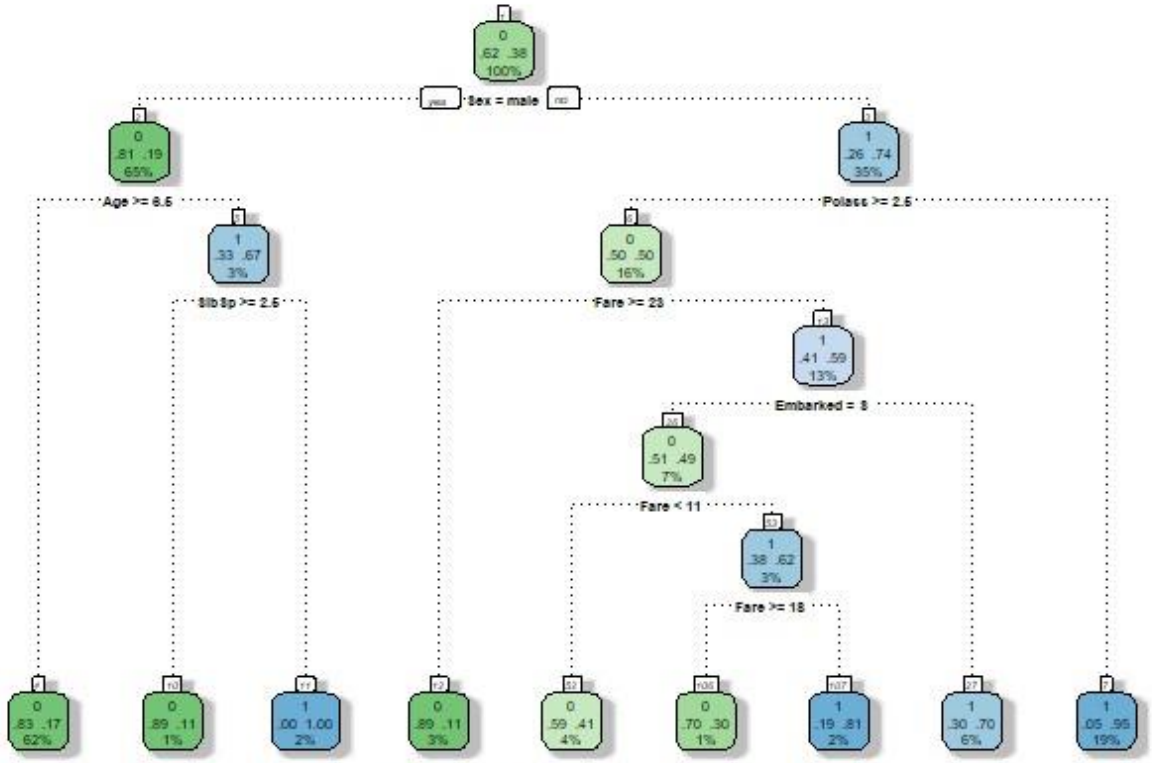


install.packages('rattle')
install.packages('rpart.plot')
install.packages('RColorBrewer')

```
> library(rpart.plot)
> fit <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, data=titanic, method="class")
> plot(fit)
> text(fit)
Warning message:
In labels.rpart(x, minlength = minlength) :
  more than 52 levels in a predicting factor, truncated for printout
```



```
> library(rattle)
> library(rpart.plot)
> library(RColorBrewer)
> fancyRpartPlot(fit)
> Prediction <- predict(fit, titanic, type="class")
> Prediction
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  0   1   1   1   0   0   0   0   1   1   1   1   0   0   1   1   0   0   0
 20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
  1   0   0   1   1   0   1   0   0   1   0   0   1   1   0   0   0   0   0
 39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
  0   1   0   1   0   1   1   0   0   1   0   0   0   0   1   1   0   1   1
 58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
  0   1   0   0   1   0   0   0   1   1   0   1   0   0   0   0   0   1   0
 77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
  0   0   1   1   0   0   1   0   1   1   0   0   1   0   0   0   0   0   0
 96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
  0   0   1   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
  0   0   0   0   0   0   0   0   0   1   0   1   0   1   1   0   0   0   1
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152
  1   0   0   1   0   0   0   1   1   1   0   0   0   0   0   0   0   0   1
153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
  0   0   0   0   1   0   0   0   0   1   0   0   0   1   1   0   0   1   0
172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
  0   1   0   0   0   0   0   0   0   0   0   1   1   1   0   1   0   0   0
191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
  1   0   1   0   1   1   0   0   1   1   0   0   0   0   0   0   0   0   1
210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228
  1   0   1   0   0   0   1   1   0   1   0   0   0   0   0   1   0   0   0
229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247
  0   0   1   0   0   1   0   1   0   1   0   0   0   1   0   0   0   1   0

  0   0   1   0   1   0   0   1   0   0   1   1   0   0   0   0   0   0   1
609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627
  1   1   0   0   1   0   0   1   0   1   1   0   0   1   1   0   0   0   0
628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646
  1   0   0   1   0   1   0   0   1   0   0   0   0   0   1   0   1   1   1
647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665
  0   1   0   1   0   1   0   1   0   0   0   1   0   0   1   0   0   0   0
666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684
  0   0   0   0   1   1   0   0   0   0   0   0   1   0   1   0   1   0   0
685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703
  1   0   0   0   0   1   1   1   1   0   0   0   0   1   1   0   1   1   0
704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722
  0   0   0   1   1   0   1   1   0   0   0   0   0   1   1   0   0   1   0
723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741
  0   0   0   0   1   1   0   1   1   0   0   0   0   0   0   1   0   0   1
742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760
  0   1   0   0   0   0   1   0   0   1   1   0   0   1   0   0   0   0   1
761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779
  0   0   0   1   0   1   0   1   0   0   0   0   1   0   1   0   0   1   0
780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798
  1   1   1   1   0   0   0   1   0   0   0   0   0   0   0   0   0   1   1
799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817
  0   0   0   1   1   1   0   0   0   0   0   1   0   0   0   0   0   0   1
818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836
  0   0   0   1   0   0   1   0   0   1   0   0   1   0   1   0   0   0   1
837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855
  0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0   1   1   1
856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874
  1   1   0   1   0   0   0   1   0   0   1   1   0   0   1   0   1   0   0
875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891
  1   1   0   0   0   1   1   0   0   0   0   0   0   1   0   1   0
Levels: 0 1
```



Rattle 2019-Mar-22 10:39:51 15DCS37