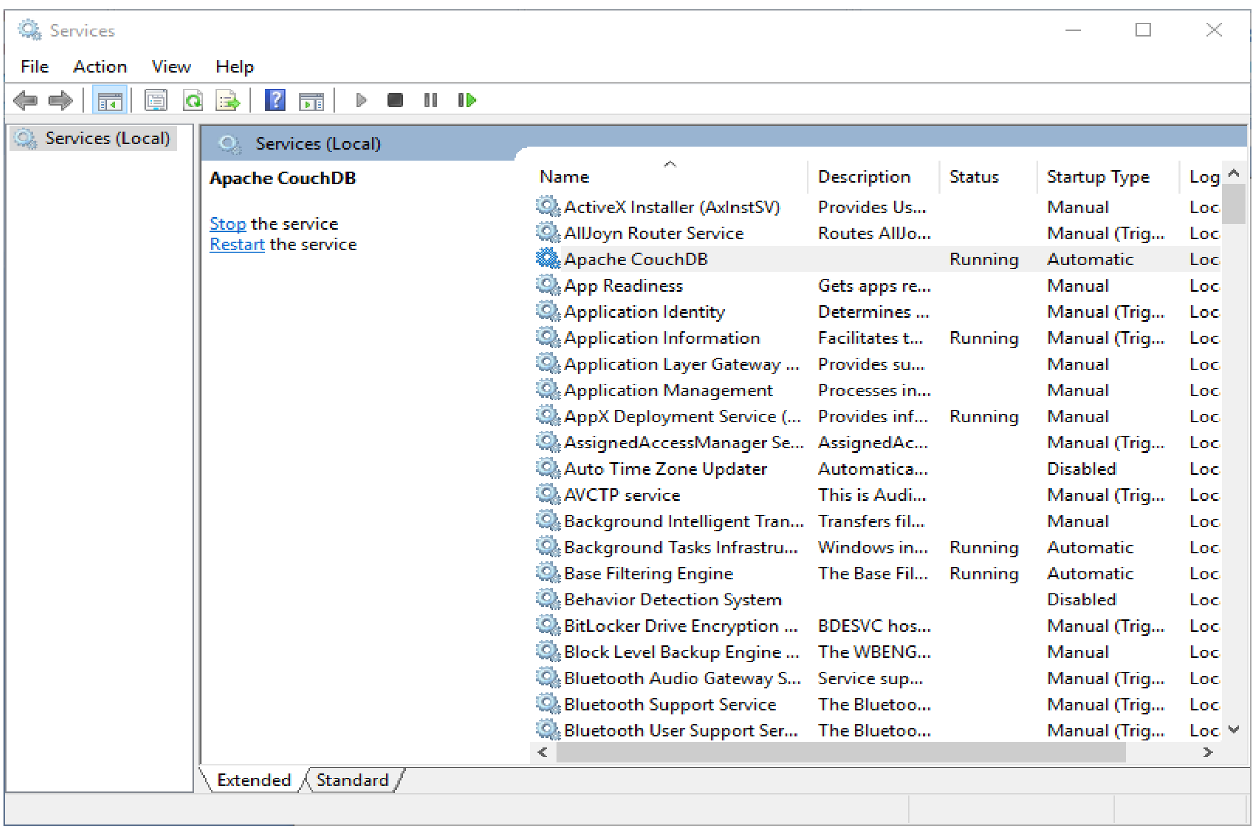
Assessment No. 01

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

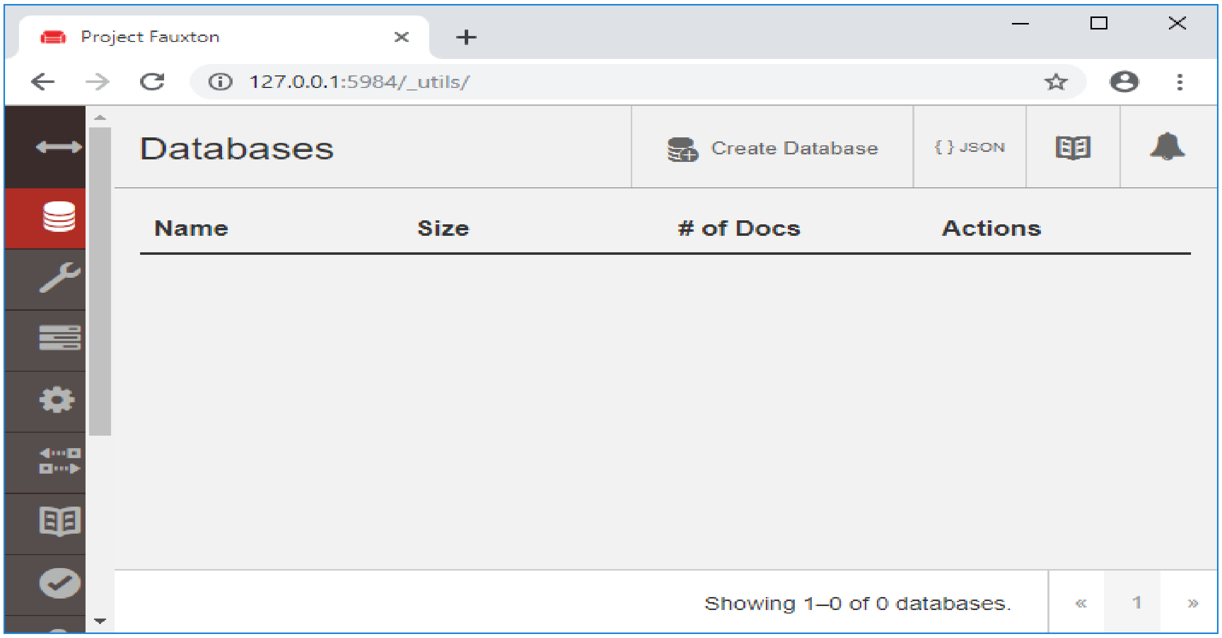
CouchDB is an open source database developed by Apache software foundation. The focus is on the ease of use, embracing the web. It is a NoSQL document store database.It uses JSON, to store data (documents), java script as its query language to transform the documents, http protocol for API to access the documents, query the indices with the web browser. It is a multi-master application released in 2005 and it became an apache project in 2008.
Semi-structured data is also an important element of many NoSQL (“not only SQL”) databases. NoSQL databases differ from relational databases because they do not separate the organization (schema) from the data.
This makes NoSQL a better choice to store information that does not easily fit into the record and table format, such as text with varying lengths.
It also allows for easier data exchange between databases.
Some newer NoSQL databases like MongoDB and Couchbase also incorporate semi-structured documents by natively storing them in the JSON format.

Step 1: Install Couchdb

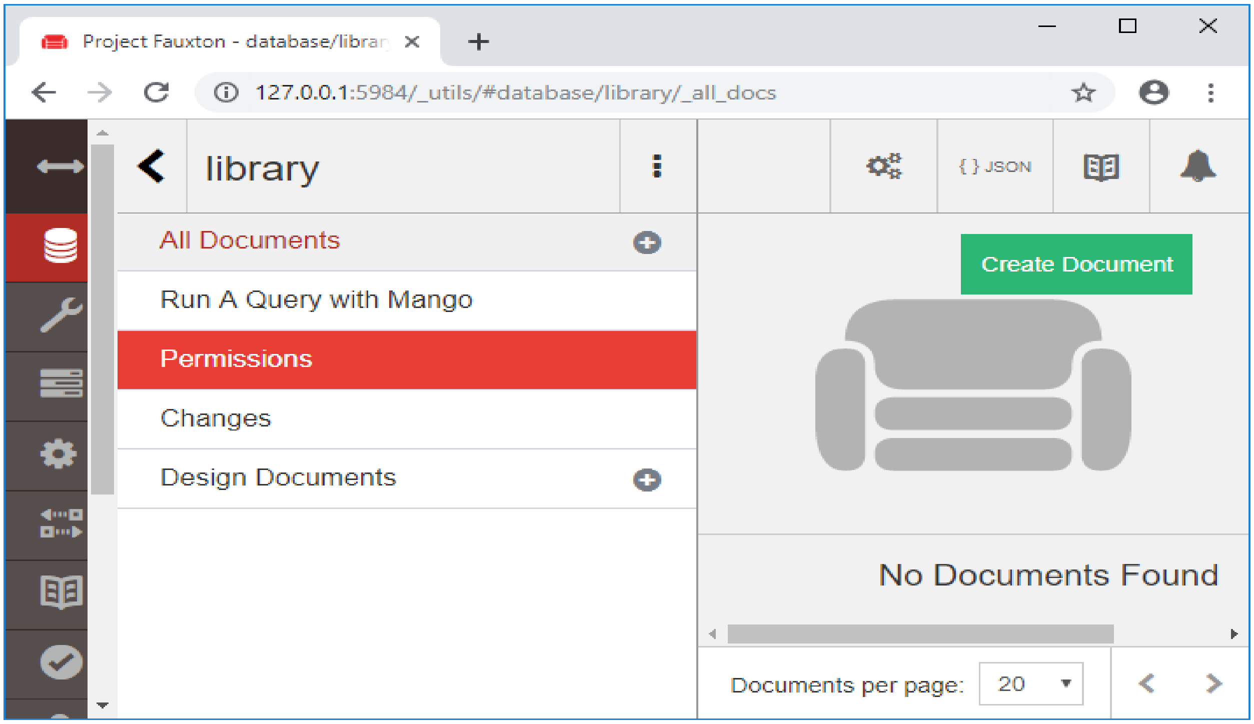
Step 2: Go to program and search “**services**” and check whether the apache couchdb in running or not.



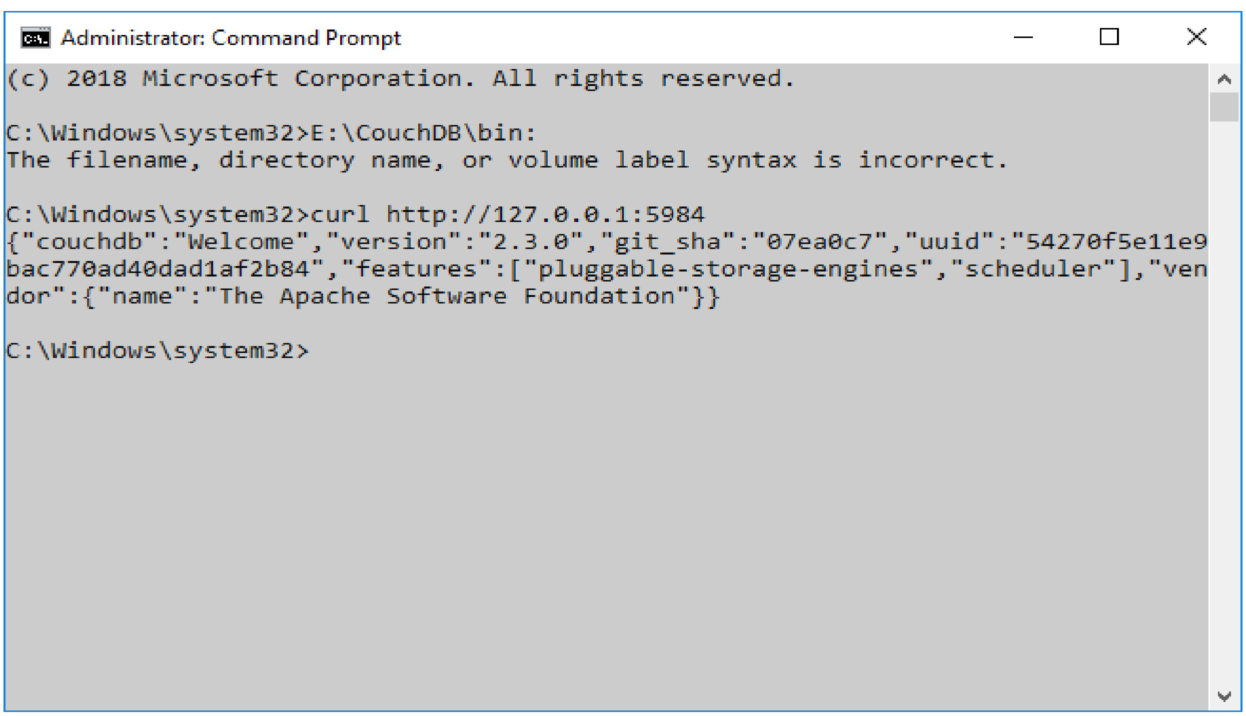
Step 3: Open browser and search **http://127.0.0.1:5984/_utils/**



Step 4: Create database directly click on “create database“>>databasename”library”.

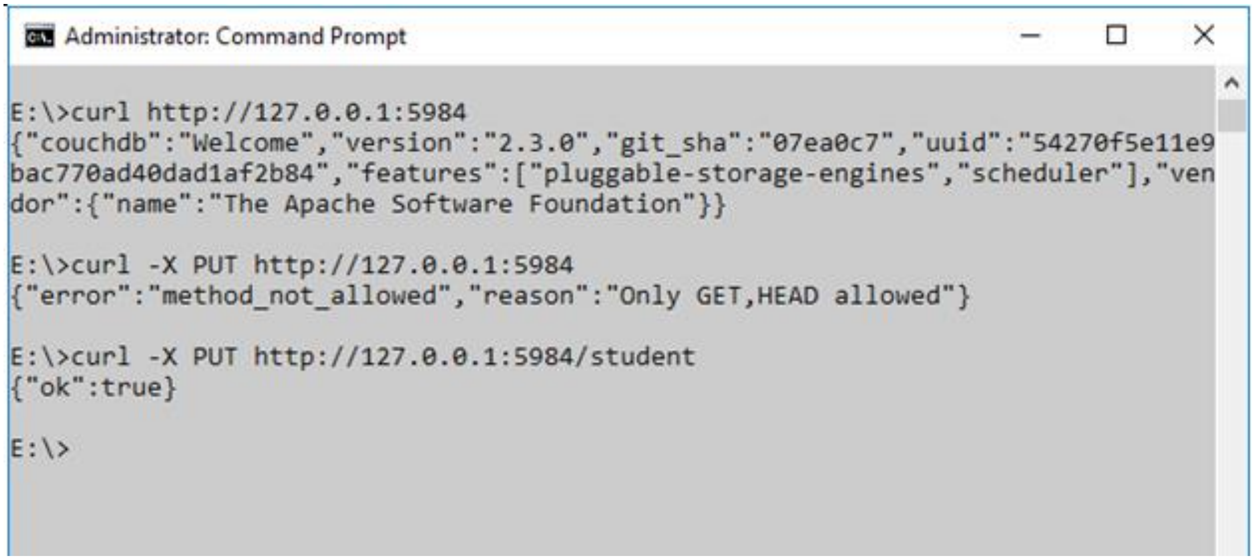


Step 5: Open the CMD as administrator mode and type curl http://127.0.0.1:5984

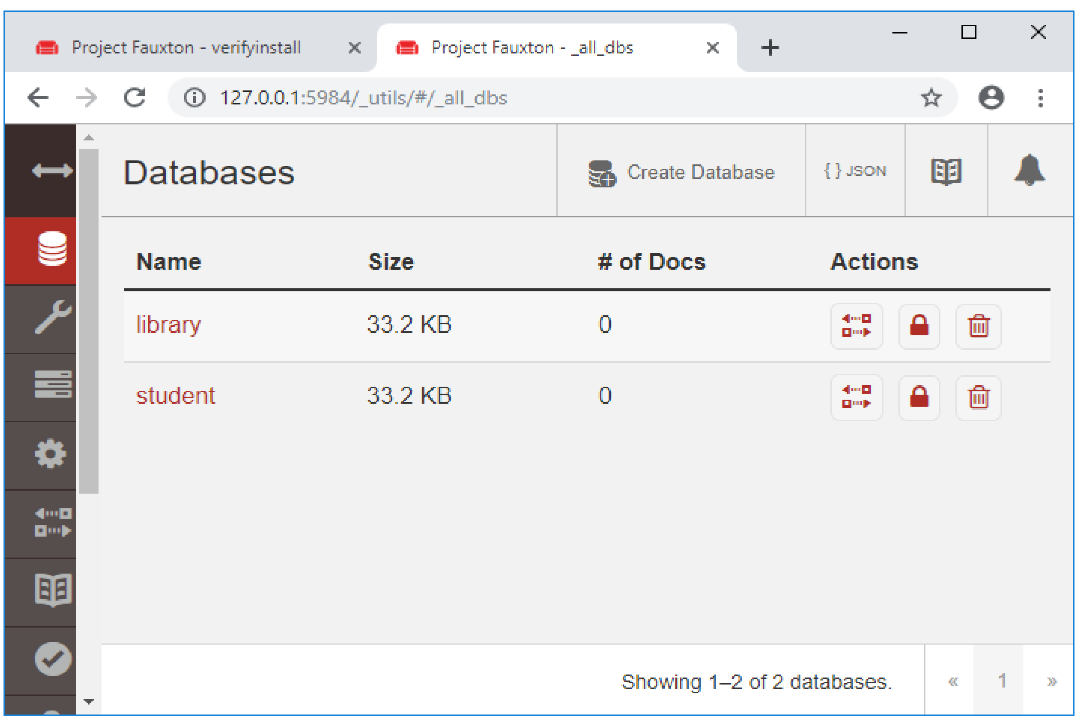


Step 6: Create student table into the command prompt type

curl -X PUT http://127.0.0.1:5984/student



Step 7: Go to the URL and see the“student” database.

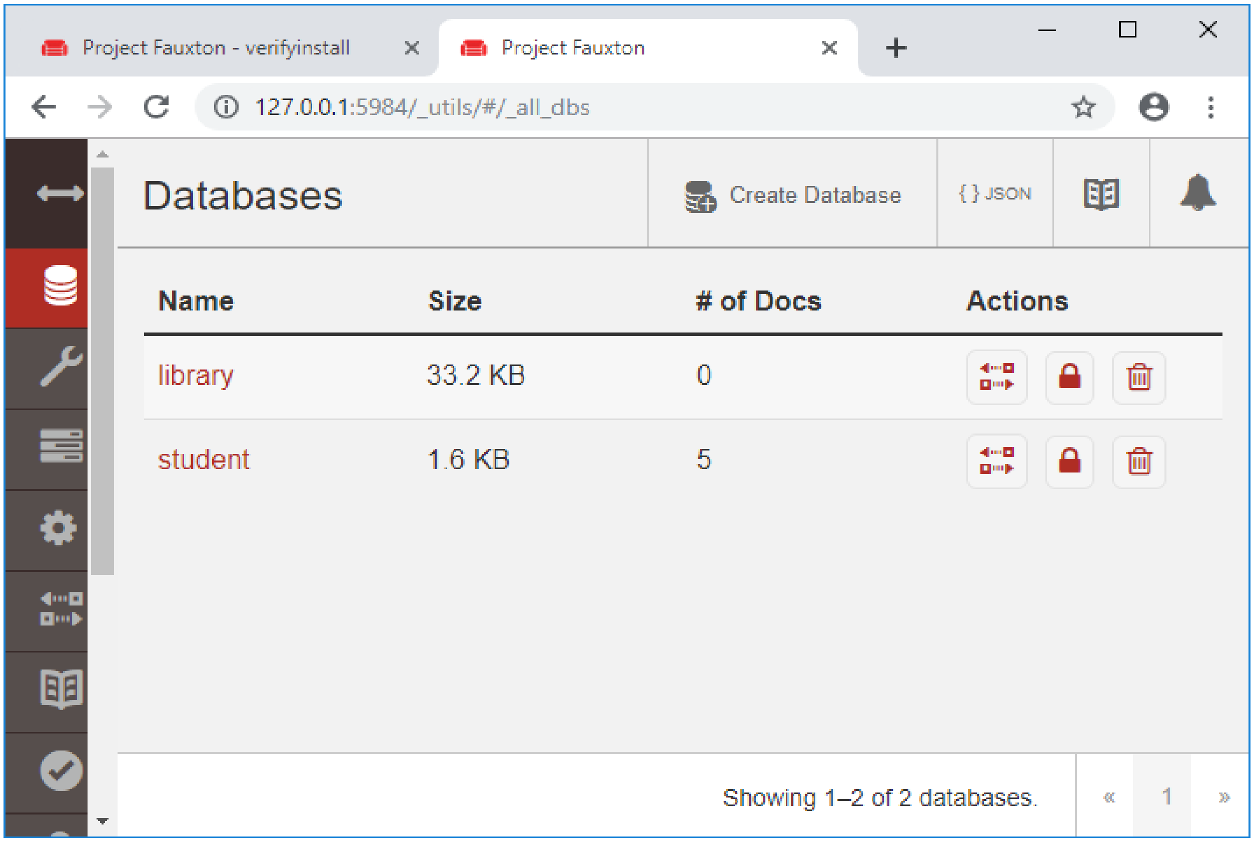


curl -X GET http://127.0.0.1:5984/_all_dbs

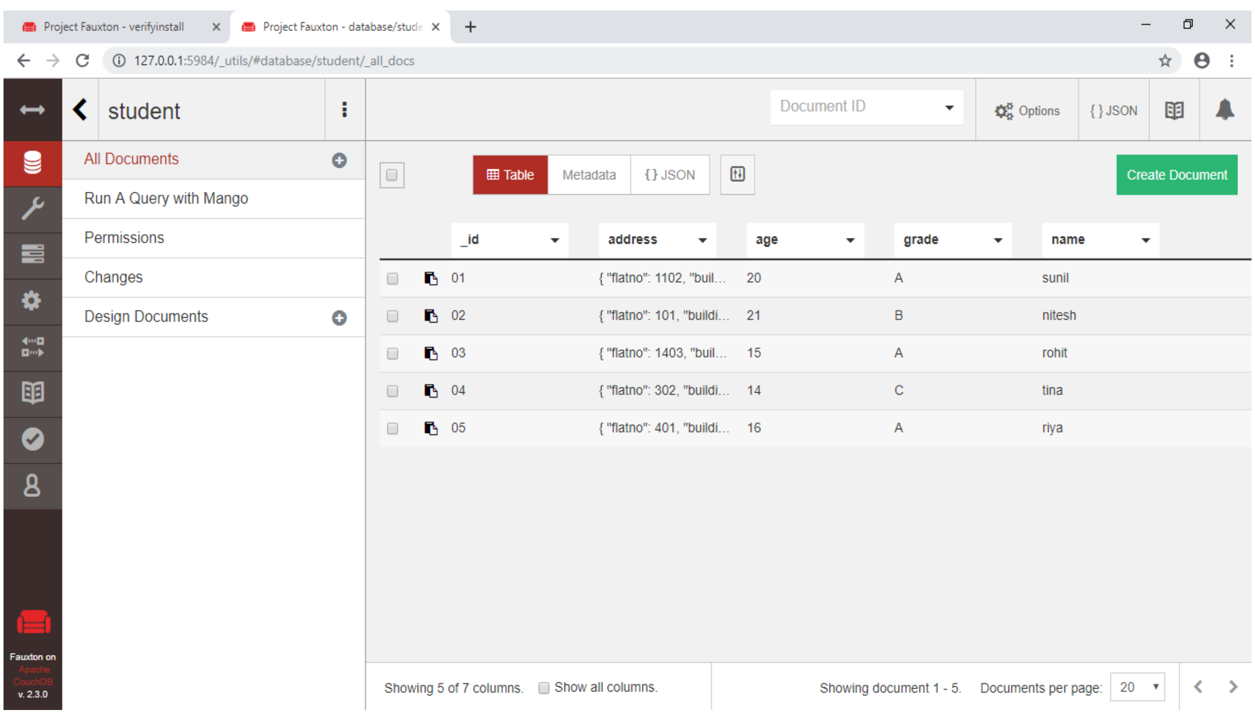
```
Administrator: Command Prompt

E:\>curl http://127.0.0.1:5984
{"couchdb":"Welcome","version":"2.3.0","git_sha":"07ea0c7","uuid":"54270f5e11e9
bac770ad40dad1af2b84","features":["pluggable-storage-engines","scheduler"],"ven
dor":{"name":"The Apache Software Foundation"}}

E:\>curl -X PUT http://127.0.0.1:5984
{"error":"method_not_allowed","reason":"Only GET,HEAD allowed"}

E:\>curl -X PUT http://127.0.0.1:5984/student
{"ok":true}

E:\>curl -X GET http://127.0.0.1:5984/_all_dbs
["library","student"]

E:\>
```

Step 8: Insert the record into the “student” database

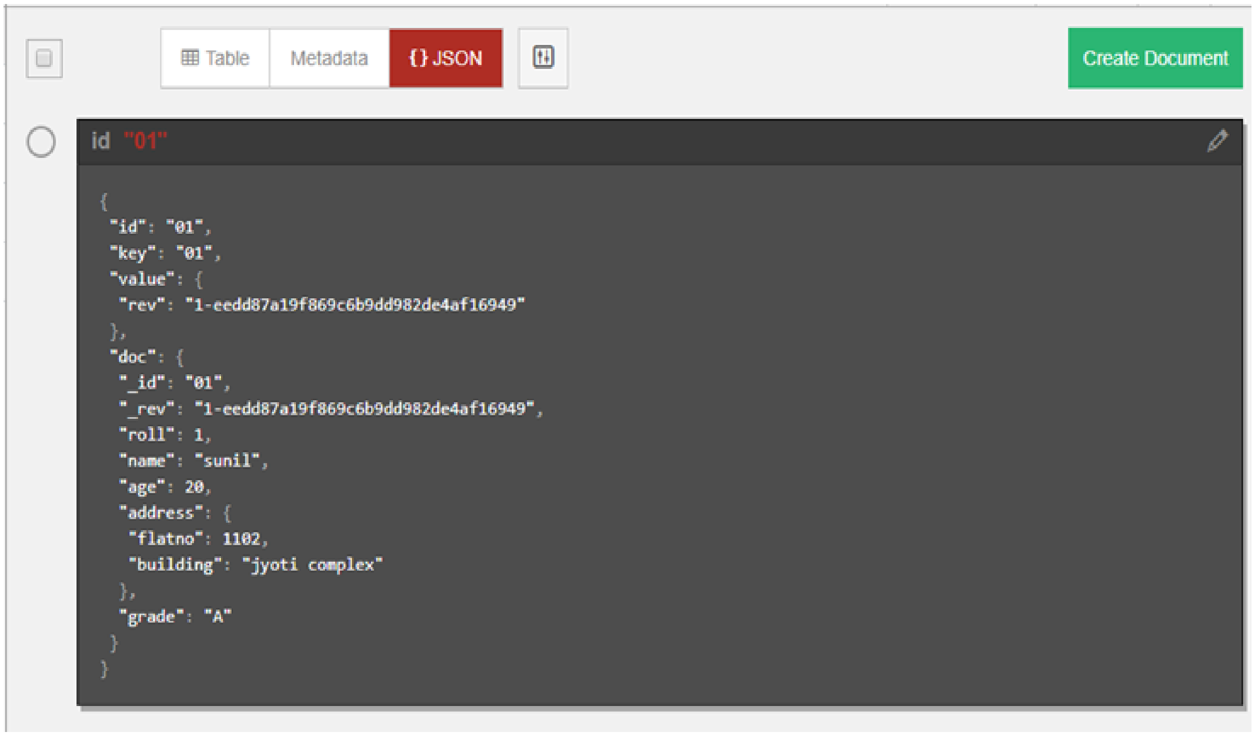
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>curl -X PUT "http://127.0.0.1:5984/student/01" -d "{\"roll\
":1,\"name\":\"sunil\",\"age\":20,\"address\":{\"flatno\":1102,\"building\":\"j
yoti complex\"},\"grade\":\"A\"}"
{"ok":true,"id":"01","rev":"1-eedd87a19f869c6b9dd982de4af16949"}

C:\Windows\system32>curl -X PUT "http://127.0.0.1:5984/student/02" -d "{\"roll\
":2,\"name\":\"nitesh\",\"age\":21,\"address\":{\"flatno\":101,\"building\":\"s
aphire hight\"},\"grade\":\"B\"}"
{"ok":true,"id":"02","rev":"1-abf0ae6fe888335baf2bf9d66f728bcf"}

C:\Windows\system32>curl -X PUT "http://127.0.0.1:5984/student/03" -d "{\"roll\
":3,\"name\":\"rohit\",\"age\":15,\"address\":{\"flatno\":1403,\"building\":\"s
pring leaf\"},\"grade\":\"A\"}"
{"ok":true,"id":"03","rev":"1-268063ac5e72387d5fc31da32f3a7c0f"}

C:\Windows\system32>curl -X PUT "http://127.0.0.1:5984/student/04" -d "{\"roll\
":4,\"name\":\"tina\",\"age\":14,\"address\":{\"flatno\":302,\"building\":\"gar
den green\"},\"grade\":\"C\"}"
{"ok":true,"id":"04","rev":"1-9bd5aff84d8526c85ec287adf2c9865c"}

C:\Windows\system32>curl -X PUT "http://127.0.0.1:5984/student/05" -d "{\"roll\
":5,\"name\":\"riya\",\"age\":16,\"address\":{\"flatno\":401,\"building\":\"rah
eja woods\"},\"grade\":\"A\"}"
{"ok":true,"id":"05","rev":"1-310f6febc018f6ea7694b0bf8f9aa0ae"}

C:\Windows\system32>
```



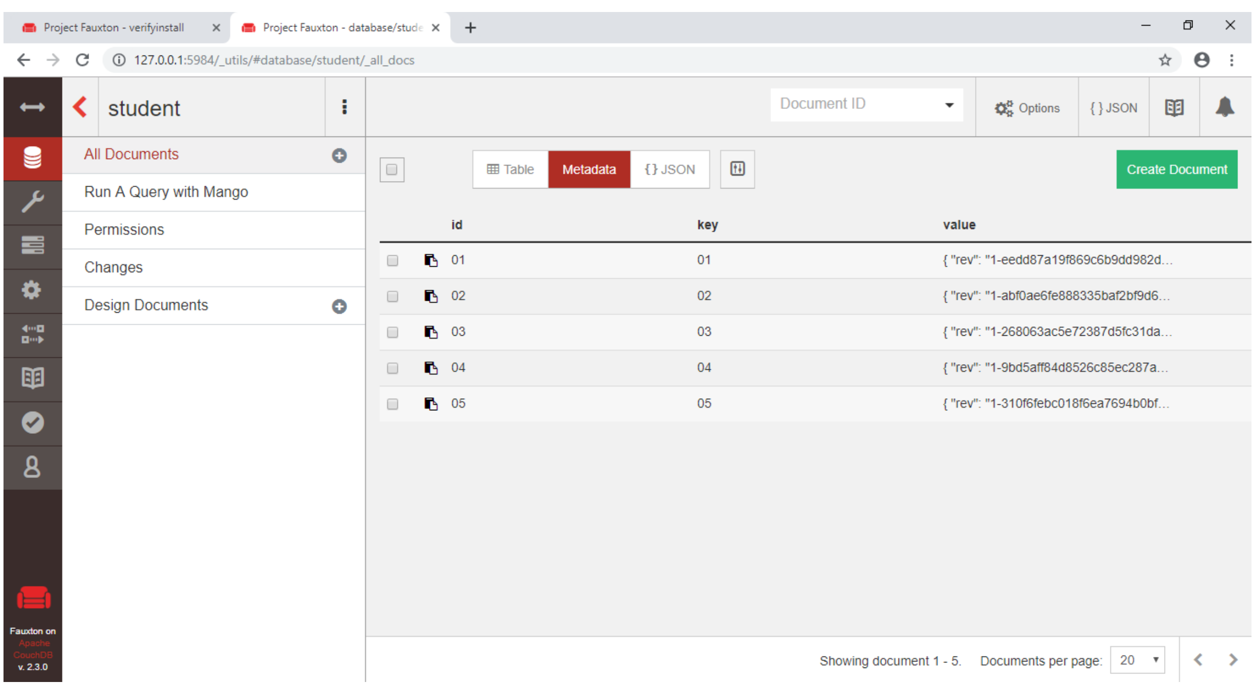
#DATA INTO TABLE FORM



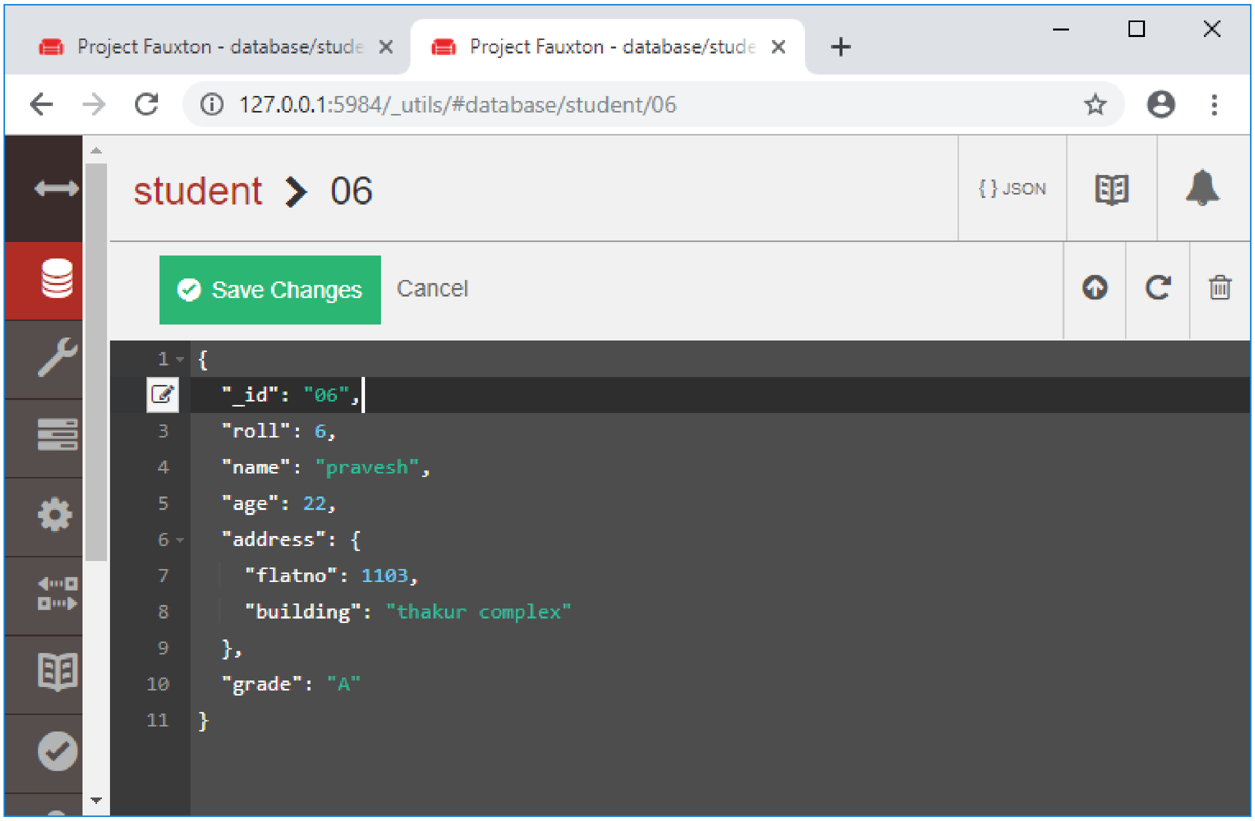
#DATA INTO JSON FORM



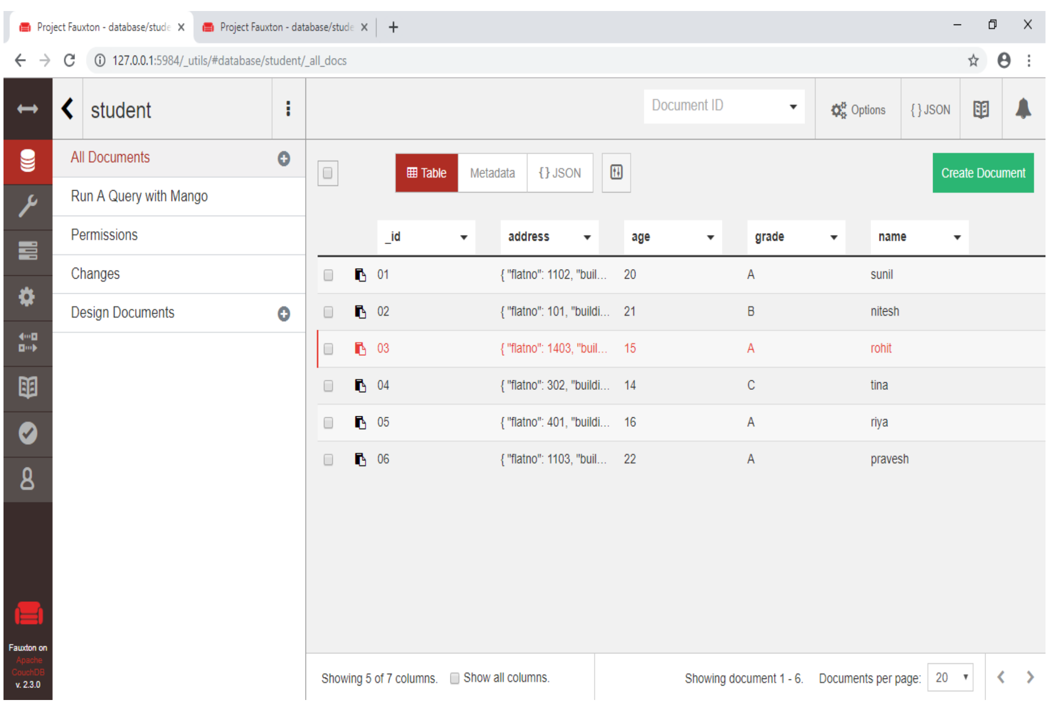
#DATA INTO METADATA FORM



Step 9: Inserts the data by the JSON

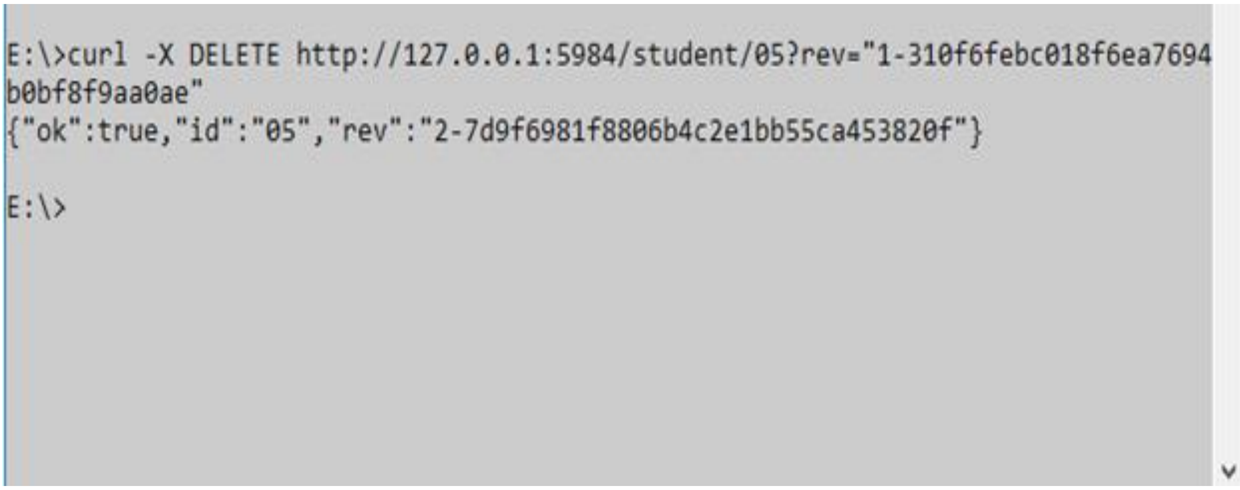


View the inserted data



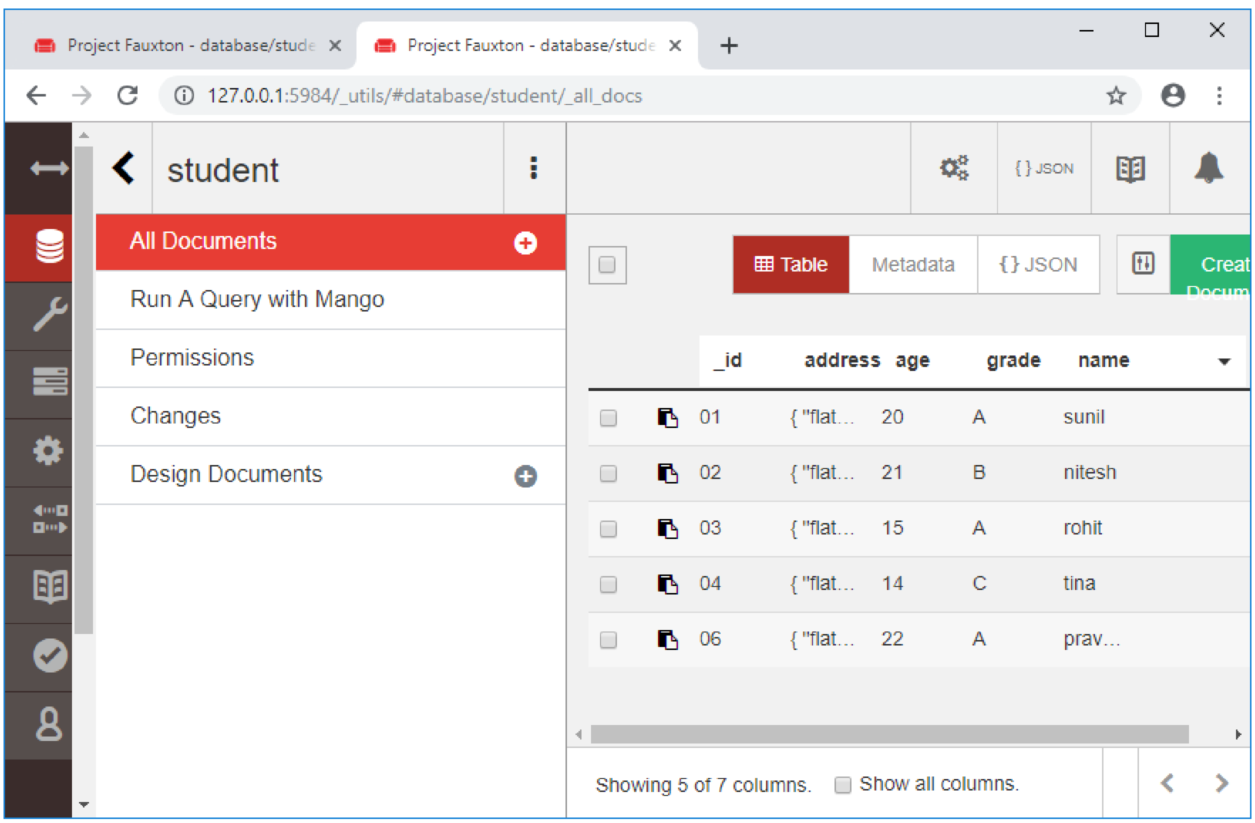
Step 10: Delete the inserteddata type from Command line

 # curl -X DELETE [http://127.0.0.1:5984/student/05? rev="1-310f6febc018f6ea7694b0bf8f9aa0ae](http://127.0.0.1:5984/student/05?rev="1-310f6febc018f6ea7694b0bf8f9aa0ae)



```
E:\>curl -X DELETE http://127.0.0.1:5984/student/05?rev="1-310f6febc018f6ea7694
b0bf8f9aa0ae"
{"ok":true,"id":"05","rev":"2-7d9f6981f8806b4c2e1bb55ca453820f"}

E:\>
```

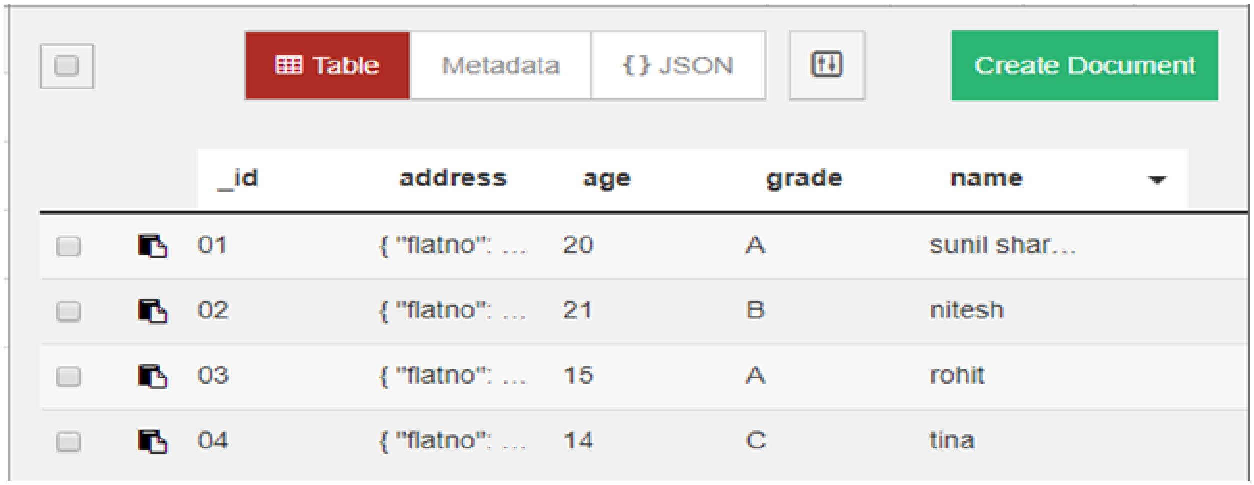
 # Here we can see that row number 5 has been deleted successfully.



Step 11: **update the student table**

```
C:\Windows\system32>curl -X PUT "http://127.0.0.1:5984/student/01" -d "{\"_rev\
":\"1-eedd87a19f869c6b9dd982de4af16949\",\"roll\":1,\"name\":\"sunil\",\"age\":
20,\"address\":{\"flatno\":1102,\"building\":\"jyoti complex\"},\"grade\":\"A\"
}
{"ok":true,"id":"01","rev":"2-915a340aba64f30d3059947f55973d8d"}
```

 # Here we can see that the first row has been successfully updated (sunil to sunilsharma).

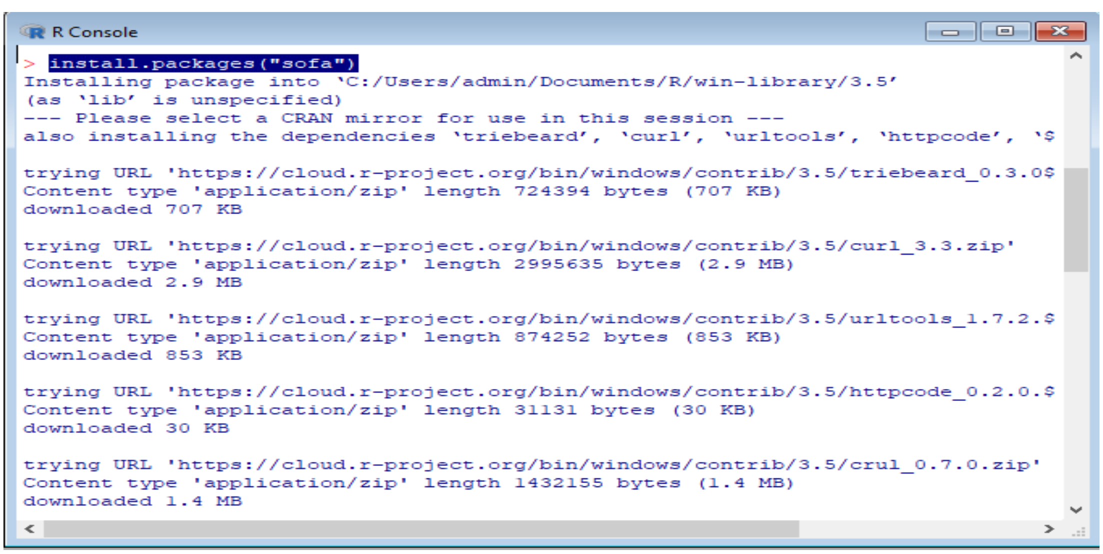


_id	address	age	grade	name
01	{ "flatno": …	20	A	sunil shar…
02	{ "flatno": …	21	B	nitesh
03	{ "flatno": …	15	A	rohit
04	{ "flatno": …	14	C	tina

Importing CouchDB Data in R with the help of package “sofa”

Step 12: Now open R console and install “sofa ” packages

>install.packages("sofa")



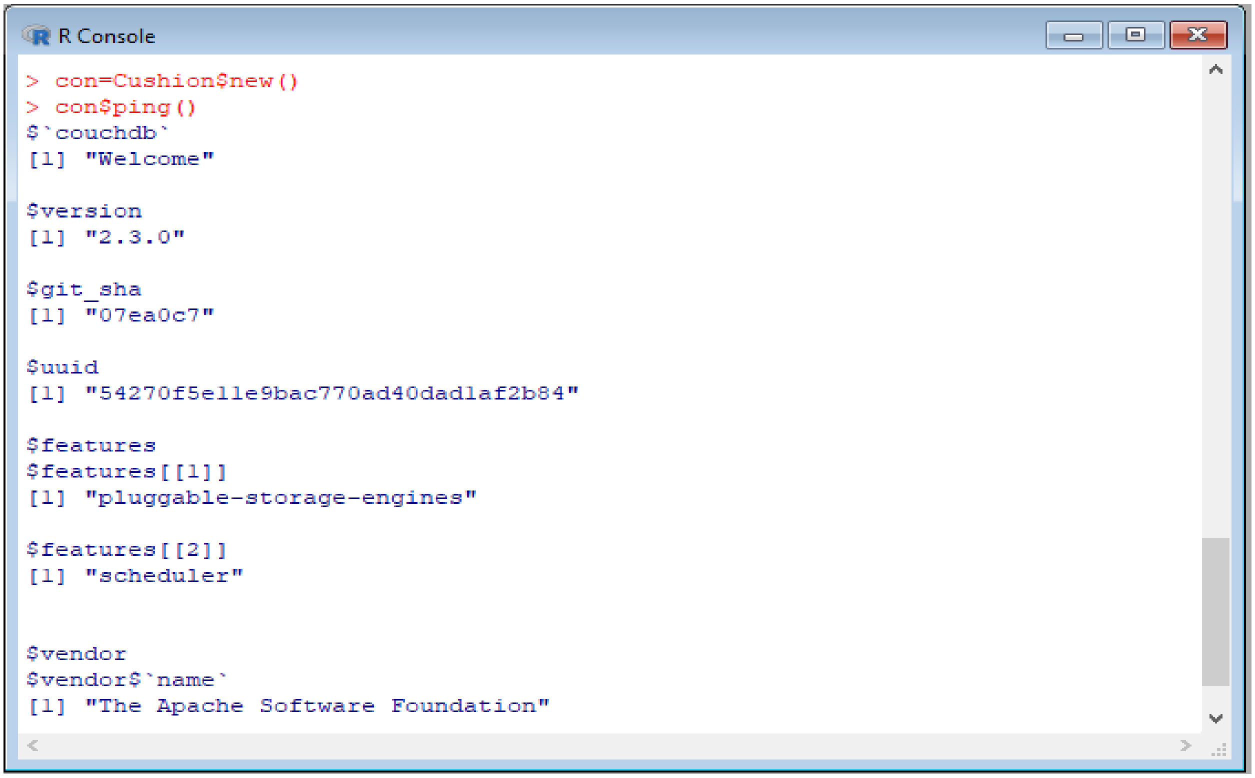
```
R Console
> install.packages("sofa")
Installing package into 'C:/Users/admin/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
also installing the dependencies 'triebeard', 'curl', 'urltools', 'httpcode', '$

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.5/triebeard_0.3.0$
Content type 'application/zip' length 724394 bytes (707 KB)
downloaded 707 KB

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.5/curl_3.3.zip'
Content type 'application/zip' length 2995635 bytes (2.9 MB)
downloaded 2.9 MB

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.5/urltools_1.7.2.$
Content type 'application/zip' length 874252 bytes (853 KB)
downloaded 853 KB

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.5/httpcode_0.2.0.$
Content type 'application/zip' length 31131 bytes (30 KB)
downloaded 30 KB

trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.5/crul_0.7.0.zip'
Content type 'application/zip' length 1432155 bytes (1.4 MB)
downloaded 1.4 MB
```

Creating a new connection

> con=Cushion$new()

Testing the new connection

>con$ping()



```
> con=Cushion$new()
> con$ping()
$`couchdb`
[1] "Welcome"

$version
[1] "2.3.0"

$git_sha
[1] "07ea0c7"

$uuid
[1] "54270f5e11e9bac770ad40dad1af2b84"

$features
$features[[1]]
[1] "pluggable-storage-engines"

$features[[2]]
[1] "scheduler"


$vendor
$vendor$`name`
[1] "The Apache Software Foundation"
```

>db_query(con,dbname="student",selector=list("_id"=list("$gt"=NULL)))$docs

```
R Console
> db_query(con,dbname="student",selector=list("_id"=list("$gt"=NULL)))$docs
[[1]]
[[1]]$`_id`
[1] "01"

[[1]]$`_rev`
[1] "2-915a340aba64f30d3059947f55973d8d"

[[1]]$roll
[1] 1

[[1]]$name
[1] "sunil"

[[1]]$age
[1] 20

[[1]]$address
[[1]]$address$`flatno`
[1] 1102

[[1]]$address$building
[1] "jyoti complex"
```

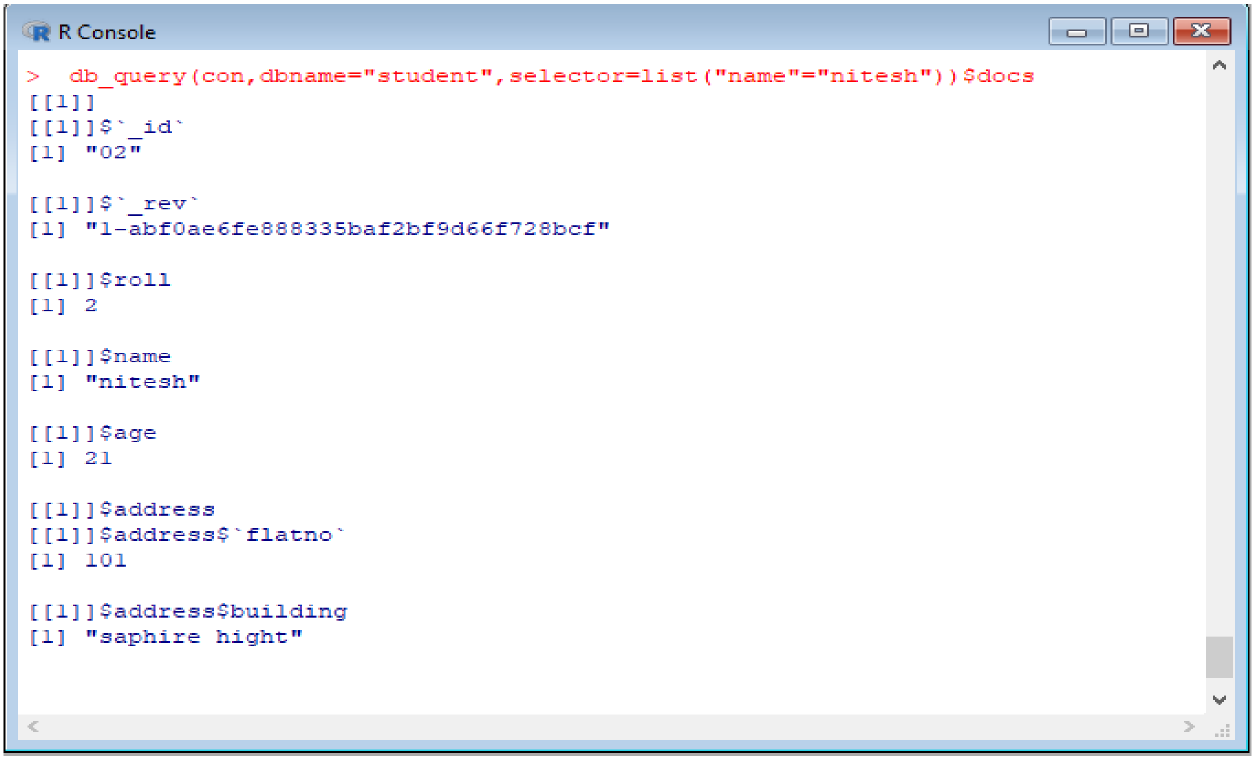
To print the students details whose age is greater than 15

>db_query(con,dbname="student",selector=list("age"=list("$gt"=15)))$docs

```
R Console
> db_query(con,dbname="student",selector=list("age"=list("$gt"=15)))$docs
[[1]]
[[1]]$`_id`
[1] "01"

[[1]]$`_rev`
[1] "2-915a340aba64f30d3059947f55973d8d"

[[1]]$roll
[1] 1

[[1]]$name
[1] "sunil"

[[1]]$age
[1] 20

[[1]]$address
[[1]]$address$`flatno`
[1] 1102

[[1]]$address$building
[1] "jyoti complex"
```

To print the details of the student with name Nitesh

>db_query(con,dbname="student",selector=list("name"="nitesh"))$docs




```
> db_query(con,dbname="student",selector=list("name"="nitesh"))$docs
[[1]]
[[1]]$`_id`
[1] "02"

[[1]]$`_rev`
[1] "1-abf0ae6fe888335baf2bf9d66f728bcf"

[[1]]$roll
[1] 2

[[1]]$name
[1] "nitesh"

[[1]]$age
[1] 21

[[1]]$address
[[1]]$address$`flatno`
[1] 101

[[1]]$address$building
[1] "saphire hight"
```

To print the details of the students who stay in building Saphire Height

>db_query(con,dbname="student",selector=list("address.building"="saphire height"))$docs

```
> db_query(con,dbname="student",selector=list("address.building"="saphire hight"))$docs
[[1]]
[[1]]$`_id`
[1] "02"

[[1]]$`_rev`
[1] "1-abf0ae6fe888335baf2bf9d66f728bcf"

[[1]]$roll
[1] 2

[[1]]$name
[1] "nitesh"

[[1]]$age
[1] 21

[[1]]$address
[[1]]$address$`flatno`
[1] 101

[[1]]$address$building
[1] "saphire hight"
```

CONCLUSION: Thus we have implemented of Data collection, Data curation and management for unstructured data.

Assessment No. 02

Aim: Practical of Data Collection, Data Curation and Management for large scale data system (such as MongoDB).

SOFTWARE USED: Mongo DB

THEORY:

Data collection

The basic **principles of data collection** include keeping things as simple as possible; planning the entire process of **data** selection, **collection**, analysis and use from the start; and ensuring that any **data collected** is valid, reliable and credible.

Mongo DB

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License.

The best MongoDB experience. Access data directly from your frontend code, intelligently distribute data for global apps, trigger serverless functions in response to data changes, and much more.

NoSQL

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

Unstructured data

Unstructured data is information that either does not have a pre-defined data model or is not organized in a pre-defined manner. Unstructured information is typically text-heavy, but may contain data such as dates, numbers, and facts as well.

Procedure:

1] Install MongoDB

Run CMD as administrator

Start MongoDB via cmd

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17763.316]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>net start MongoDB
The requested service has already been started.

More help is available by typing NET HELPMSG 2182.


C:\WINDOWS\system32>
```

2] Enter the MongoDB bin folder via the cmd.

Type Mongo

```
Administrator: Command Prompt - mongo
C:\WINDOWS\system32>cd C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin>mongo
MongoDB shell version v4.0.6
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("fef6c64f-6507-41d6-bed2-de47c928ec45") }
MongoDB server version: 4.0.6
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-03-08T10:28:17.164+0530 I CONTROL  [initandlisten]
2019-03-08T10:28:17.165+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-03-08T10:28:17.165+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2019-03-08T10:28:17.165+0530 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

3] Check Current Database

```
> db
test
>
```

4] Use database library

```
> use library
switched to db library
>
```

5] Create Collection(table)

```
> db.createCollection("book_info")
{ "ok" : 1 }
>
```

Insert document (records) in collection

```
> db.book_info.insert({title:'Programming with Java',status_info:{accession_no:BS0001,status:ISSUES},)
...
...
> db.book_info.insert({title:'PROGRAMMING WITH JAVA',
... status_info:{accession_no:'BS0001',status:'ISSUED'},
... author:'EBALAGURUSWAMY',
... cost:'350',
... publisher:{name:'TMH',location:'banglore'}})
WriteResult({ "nInserted" : 1 })
```

To insert multiple records in one column

```
> db.book_info.insert({title:'DISTRIBUTED SYSTEMS', status_info:[{accession_no:'BS0005',status:'ISSUED'},{accession_no:
BS0006',status:'ISSUED'},{accesion_no:'BS0007',status:'ISSUED'},{accesion_no:'BS0008',status:'AVAIL'}], author:'ANDREW
TANENBAUM', cost:'350', publisher:{name:'PEARSON',location:'ANDHERI'}})
WriteResult({ "nInserted" : 1 })
```

6] To view records according the title:

```
> db.book_info.find({title:'LET US C'})
{ "_id" : ObjectId("5c8201d69cff5f2a54951b57"), "title" : "LET US C", "status_info" : { "accession_no" : "BS0009", "stat
us" : "AVAIL" }, "author" : "KANETKAR YASHWANT P", "cost" : "600", "publisher" : { "name" : "B.P.B", "location" : "RAJAS
THAN" } }
>
```

7] To view all records:

```
> db.book_info.find({})
{ "_id" : ObjectId("5c81fd569cff5f2a54951b52"), "title" : "PROGRAMMING WITH JAVA", "status_info" : { "accession_no" : "BS0001", "status" : "ISSUED" }, "author" : "EBALAGURUSWAMY", "cost" : "350", "publisher" : { "name" : "TMH", "location" : "banglore" } }
{ "_id" : ObjectId("5c81fdd89cff5f2a54951b53"), "title" : "ASP.NET3.5 VB 2008", "status_info" : { "accession_no" : "BS0002", "status" : "AVAIL" }, "author" : "ANNE BOEHM", "cost" : "650", "publisher" : { "name" : "MURACH", "location" : "JAIPUR" } }
{ "_id" : ObjectId("5c81fe5e9cff5f2a54951b54"), "title" : "PRPGRAMMING IN VB", "status_info" : { "accession_no" : "BS0003", "status" : "ISSUED" }, "author" : "JULIA CASE BRADLEY", "cost" : "600", "publisher" : { "name" : "TMH", "location" : "BANGLORE" } }
{ "_id" : ObjectId("5c81feaf9cff5f2a54951b55"), "title" : "DATABASE SYSTEM CONCEPTS", "status_info" : { "accession_no" : "BS0004", "status" : "ISSUED" }, "author" : "KORTH SUDARSHAN", "cost" : "500", "publisher" : { "name" : "TMH", "location" : "BANGLORE" } }
{ "_id" : ObjectId("5c82005f9cff5f2a54951b56"), "title" : "DISTRIBUTED SYSTEMS", "status_info" : [ { "accession_no" : "BS0005", "status" : "ISSUED" }, { "accession_no" : "BS0006", "status" : "ISSUED" }, { "accesion_no" : "BS0007", "status" : "ISSUED" }, { "accession_no" : "BS0008", "status" : "AVAIL" } ], "author" : "ANDREW TANENBAUM", "cost" : "350", "publisher" : { "name" : "PEARSON", "location" : "ANDHERI" } }
{ "_id" : ObjectId("5c8201d69cff5f2a54951b57"), "title" : "LET US C", "status_info" : { "accession_no" : "BS0009", "status" : "AVAIL" }, "author" : "KANETKAR YASHWANT P", "cost" : "600", "publisher" : { "name" : "B.P.B", "location" : "RAJASTHAN" } }
{ "_id" : ObjectId("5c8202f69cff5f2a54951b58"), "title" : "MODERN DIGITAL ELECTRONICS", "status_info" : [ { "accession_no" : "BS010", "status" : "ISSUED" }, { "accession_no" : "BS011", "status" : "AVAIL" } ], "author" : "JAIN R.P", "cost" : "650", "publisher" : { "name" : "TMH", "location" : "BANGLORE" } }
```

8] To view records of accession numbers:

```
> db.book_info.find({},{'status_info.accession_no':1})
{ "_id" : ObjectId("5c81fd569cff5f2a54951b52"), "status_info" : { "accession_no" : "BS0001" } }
{ "_id" : ObjectId("5c81fdd89cff5f2a54951b53"), "status_info" : { "accession_no" : "BS0002" } }
{ "_id" : ObjectId("5c81fe5e9cff5f2a54951b54"), "status_info" : { "accession_no" : "BS0003" } }
{ "_id" : ObjectId("5c81feaf9cff5f2a54951b55"), "status_info" : { "accession_no" : "BS0004" } }
{ "_id" : ObjectId("5c82005f9cff5f2a54951b56"), "status_info" : [ { "accession_no" : "BS0005" }, { "accession_no" : "BS0006" }, {  }, { "accession_no" : "BS0008" } ] }
{ "_id" : ObjectId("5c8201d69cff5f2a54951b57"), "status_info" : { "accession_no" : "BS0009" } }
{ "_id" : ObjectId("5c8202f69cff5f2a54951b58"), "status_info" : [ { "accession_no" : "BS010" }, { "accession_no" : "BS011" } ] }
```

To print records with accession no “BS0001”

```
> db.book_info.find({'status_info.accession_no':'BS0001'})
{ "_id" : ObjectId("5c81fd569cff5f2a54951b52"), "title" : "PROGRAMMING WITH JAVA", "status_info" : { "accession_no" : "B
S0001", "status" : "ISSUED" }, "author" : "EBALAGURUSWAMY", "cost" : "350", "publisher" : { "name" : "TMH", "location" :
 "banglore" } }
```

9] To print data in JSON format

Administrator: Command Prompt - mongo

```
> db.book_info.find().forEach(printjson)
{
        "_id" : ObjectId("5c81fd569cff5f2a54951b52"),
        "title" : "PROGRAMMING WITH JAVA",
        "status_info" : {
                "accession_no" : "BS0001",
                "status" : "ISSUED"
        },
        "author" : "EBALAGURUSWAMY",
        "cost" : "350",
        "publisher" : {
                "name" : "TMH",
                "location" : "banglore"
        }
}
{
        "_id" : ObjectId("5c81fdd89cff5f2a54951b53"),
        "title" : "ASP.NET3.5 VB 2008",
        "status_info" : {
                "accession_no" : "BS0002",
                "status" : "AVAIL"
        },
        "author" : "ANNE BOEHM",
        "cost" : "650",
        "publisher" : {
                "name" : "MURACH",
                "location" : "JAIPUR"
        }
}
{
```

10] To print the details of book where cost is more than 500

```
> db.book_info.find({"cost":{$gt:"500"}}).pretty()
{
        "_id" : ObjectId("5c81fdd89cff5f2a54951b53"),
        "title" : "ASP.NET3.5 VB 2008",
        "status_info" : {
                "accession_no" : "BS0002",
                "status" : "AVAIL"
        },
        "author" : "ANNE BOEHM",
        "cost" : "650",
        "publisher" : {
                "name" : "MURACH",
                "location" : "JAIPUR"
        }
}
{
        "_id" : ObjectId("5c81fe5e9cff5f2a54951b54"),
        "title" : "PRPGRAMMING IN VB",
        "status_info" : {
                "accession_no" : "BS0003",
                "status" : "ISSUED"
        },
        "author" : "JULIA CASE BRADLEY",
        "cost" : "600",
        "publisher" : {
                "name" : "TMH",
                "location" : "BANGLORE"
        }
```

Conclusion: We have successfully executed Data Collection, Data Curation and Management for large scale data system (such as MongoDB).

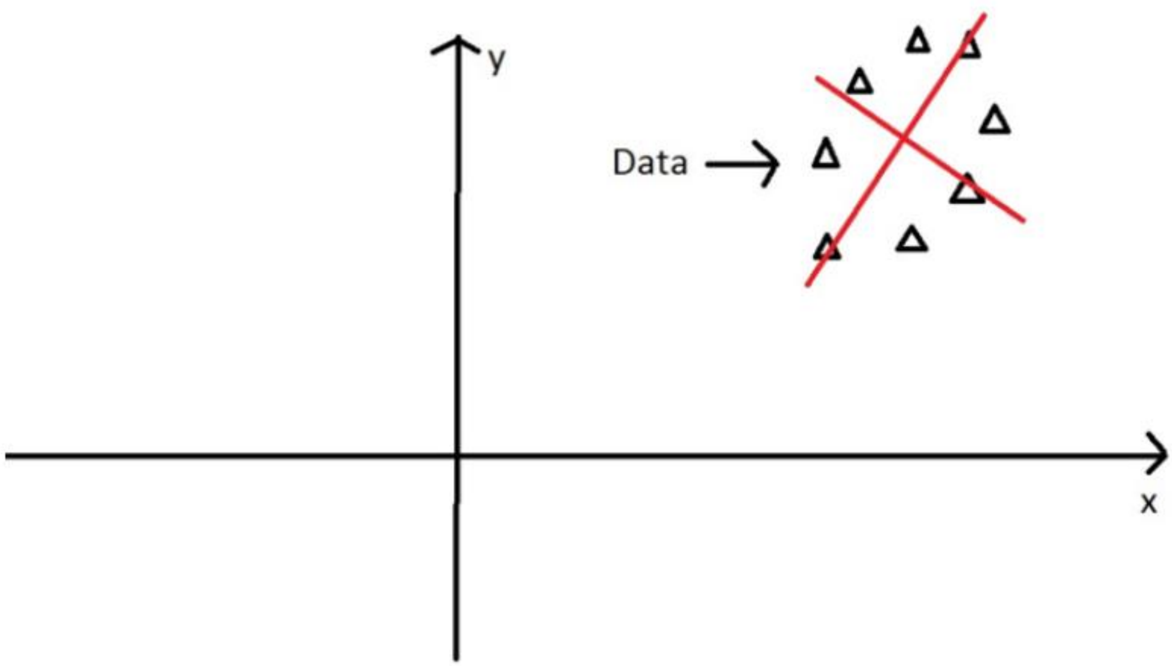
Assessment No. 03

Aim: Principal component analysis

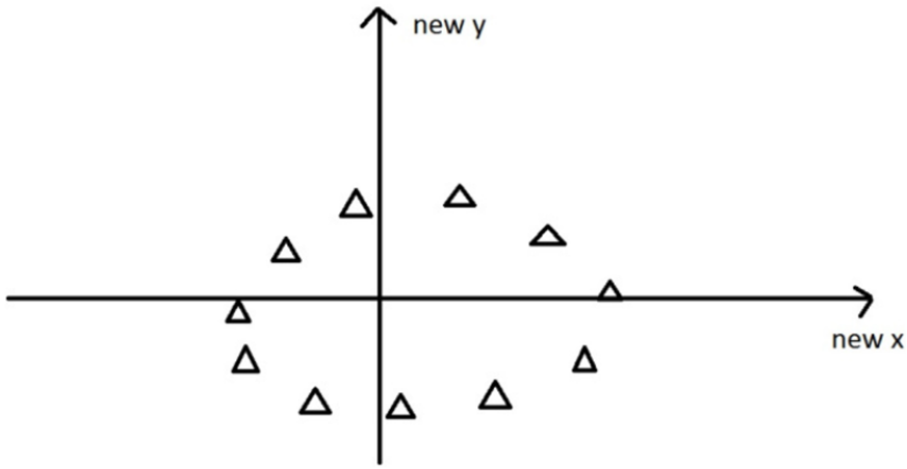
Principal components are the directions where there is the most variance, the directions where the data is most spread out. When we get a set of data points, like the triangles above, we can deconstruct the set into eigenvectors and eigenvalues. Eigenvectors and values exist in pairs: Every eigenvector has a corresponding eigenvalue. An eigenvector is a direction, and an eigenvalue is a number, telling you how much variance there is in the data in that direction, the eigenvector with the highest eigenvalue is therefore the principal component.

Eigenvalues: The numbers on **the diagonal of the diagonalized covariance matrix** are called eigenvalues of the covariance matrix. Large eigenvalues correspond to large variances.

Eigenvectors: The directions of the new rotated axes are called the eigenvectors of the covariance matrix.



Reframing data in new dimensions



Case study 1

Data: Data consists of 5 records of students in 3 different subjects.

	A	B	C	D
1	stud_roll	Math	English	Art
2	1	90	60	90
3	2	90	90	30
4	3	60	60	60
5	4	60	60	90
6	5	30	30	30
7		66	60	60
8				

Creating Covariance matrix to generate the Eigen values and Eigen Vectors in Excel

Step 1: Calculating the deviation from the mean

step 1	Math	English	Art
	24	0	30
	24	30	-30
	-6	0	0
	-6	0	30
	-36	-30	-30

Step 2: Transpose the matrix

step 2		Transposition Matrix			
	1	2	3	4	5
Math	24	24	-6	-6	-36
English	0	30	0	0	-30
Art	30	-30	0	30	-30

Step 3: Creating co variance matrix

Step 3					
V=TA.A/N					
co-variance matrix					
			630	450	225
			450	450	0
			225	0	900

<u>**Practical in R**</u>

R Console

```
> x=read.csv("d:/tycs/students.csv")
> x
  Math English Art
1   90      60  90
2   90      90  30
3   60      60  60
4   60      60  90
5   30      30  30
> cov_mat=cov(x)
> cov_mat
        Math English Art
Math     630     450 225
English  450     450   0
Art      225       0 900
```

Verifying the covariance matrix in R with the matrix generated in Excel.

R Console

```
> ex=eigen(cov_mat)
> ex
eigen() decomposition
$`values`
[1] 1137.58744  786.38798   56.02458

$vectors
          [,1]       [,2]       [,3]
[1,] 0.6558023 -0.3859988  0.6487899
[2,] 0.4291978 -0.5163664 -0.7410499
[3,] 0.6210577  0.7644414 -0.1729644
```

R Console

```
> install.packages('factoextra',repos="http://cran.us.r-project.org")
Installing package into 'C:/Users/admin/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
also installing the dependencies 'ellipsis', 'clipr', 'rematch', 'prettyunits',$

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/3.5/ellipsis_0.1.0$
Content type 'application/zip' length 32597 bytes (31 KB)
downloaded 31 KB

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/3.5/clipr_0.5.0.zi$
Content type 'application/zip' length 40746 bytes (39 KB)
downloaded 39 KB

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/3.5/rematch_1.0.1.$
Content type 'application/zip' length 16008 bytes (15 KB)
downloaded 15 KB

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/3.5/prettyunits_1.$
Content type 'application/zip' length 33084 bytes (32 KB)
downloaded 32 KB

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/3.5/forcats_0.4.0.$
Content type 'application/zip' length 344080 bytes (336 KB)
downloaded 336 KB
```

R Console

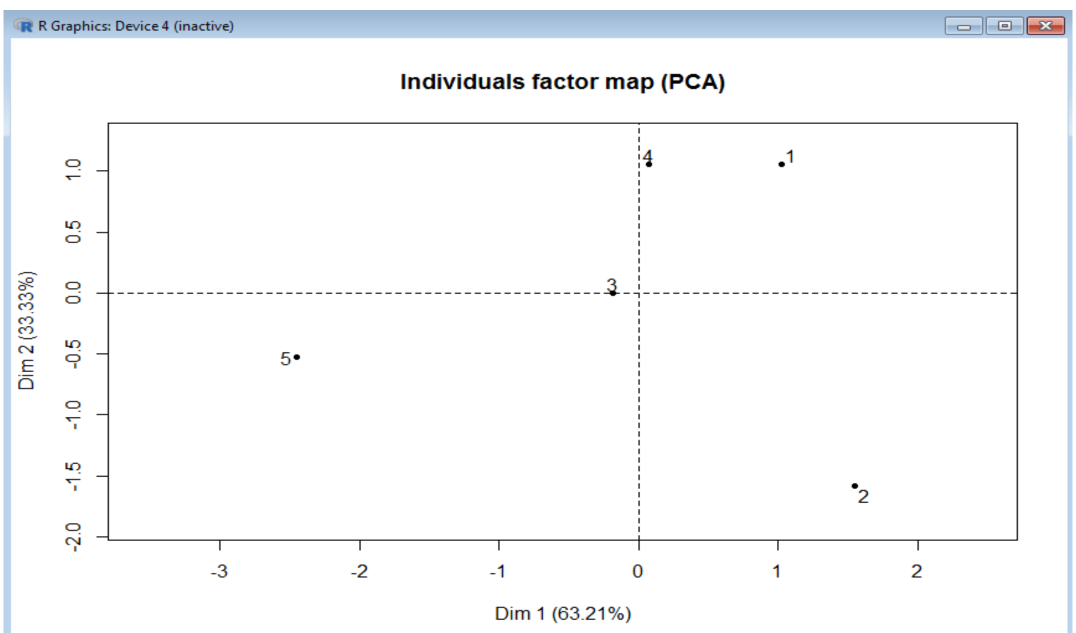
```
> library("FactoMineR")
Warning message:
package 'FactoMineR' was built under R version 3.5.2
```

Generating Principal components

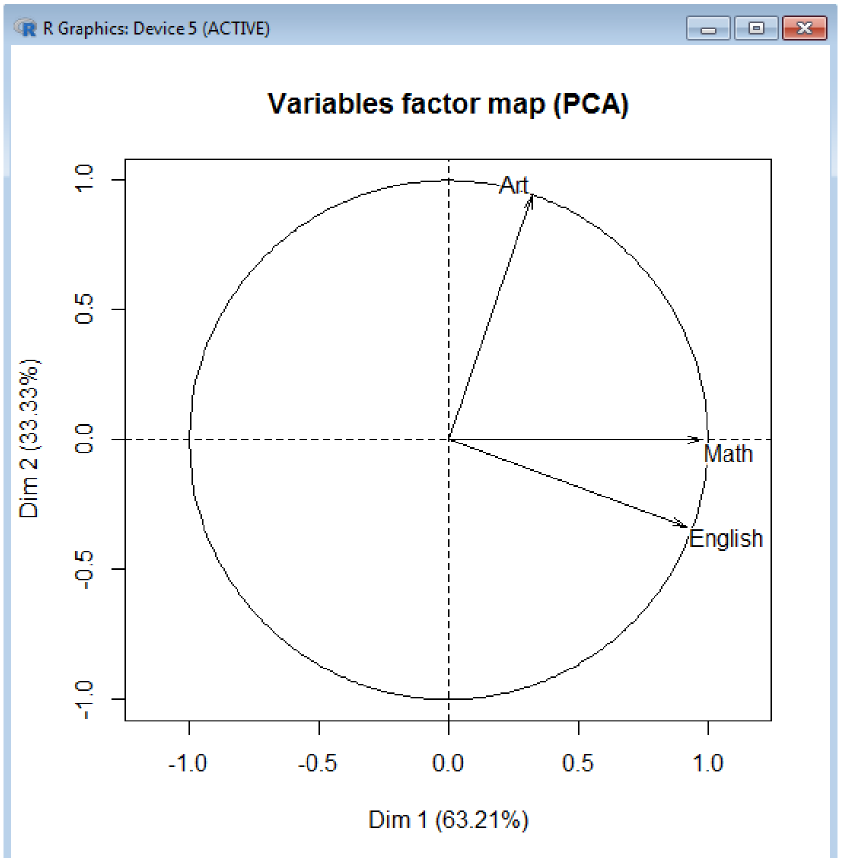
R Console

```
> datapca=PCA(x,ncp=3,graph=TRUE)
> print(datapca)
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 5 individuals, described by 3 variables
*The results are available in the following objects:

   name                 description
1  "$eig"               "eigenvalues"
2  "$var"               "results for the variables"
3  "$var$coord"         "coord. for the variables"
4  "$var$cor"           "correlations variables - dimensions"
5  "$var$cos2"          "cos2 for the variables"
6  "$var$contrib"       "contributions of the variables"
7  "$ind"               "results for the individuals"
8  "$ind$coord"         "coord. for the individuals"
9  "$ind$cos2"          "cos2 for the individuals"
10 "$ind$contrib"       "contributions of the individuals"
11 "$call"              "summary statistics"
12 "$call$centre"       "mean of the variables"
13 "$call$ecart.type"   "standard error of the variables"
14 "$call$row.w"        "weights for the individuals"
15 "$call$col.w"        "weights for the variables"
```

Position of each student on the Graph



Showing the variance of each component of the data



A **Biplot** is an enhanced scatterplot that uses both points and vectors to represent structure. A **biplot** uses points to represent the scores of the observations on the **principal components**, and it uses vectors to represent the coefficients of the variables on the **principal components**.

The angles between the vectors tell us how characteristics correlate with one another.

When two vectors are close, forming a small angle, the two variables they represent are positively correlated. Example: Math and English

- If they meet each other at 90°, they are not likely to be correlated.
- When they diverge and form a large angle (close to 180°), they are negative correlated.

R Console

```
> datapca$eig
       eigenvalue percentage of variance cumulative percentage of variance
comp 1  1.8964215              63.214049                          63.21405
comp 2  1.0000000              33.333333                          96.54738
comp 3  0.1035785               3.452618                         100.00000
> datapca$loadings
NULL
```

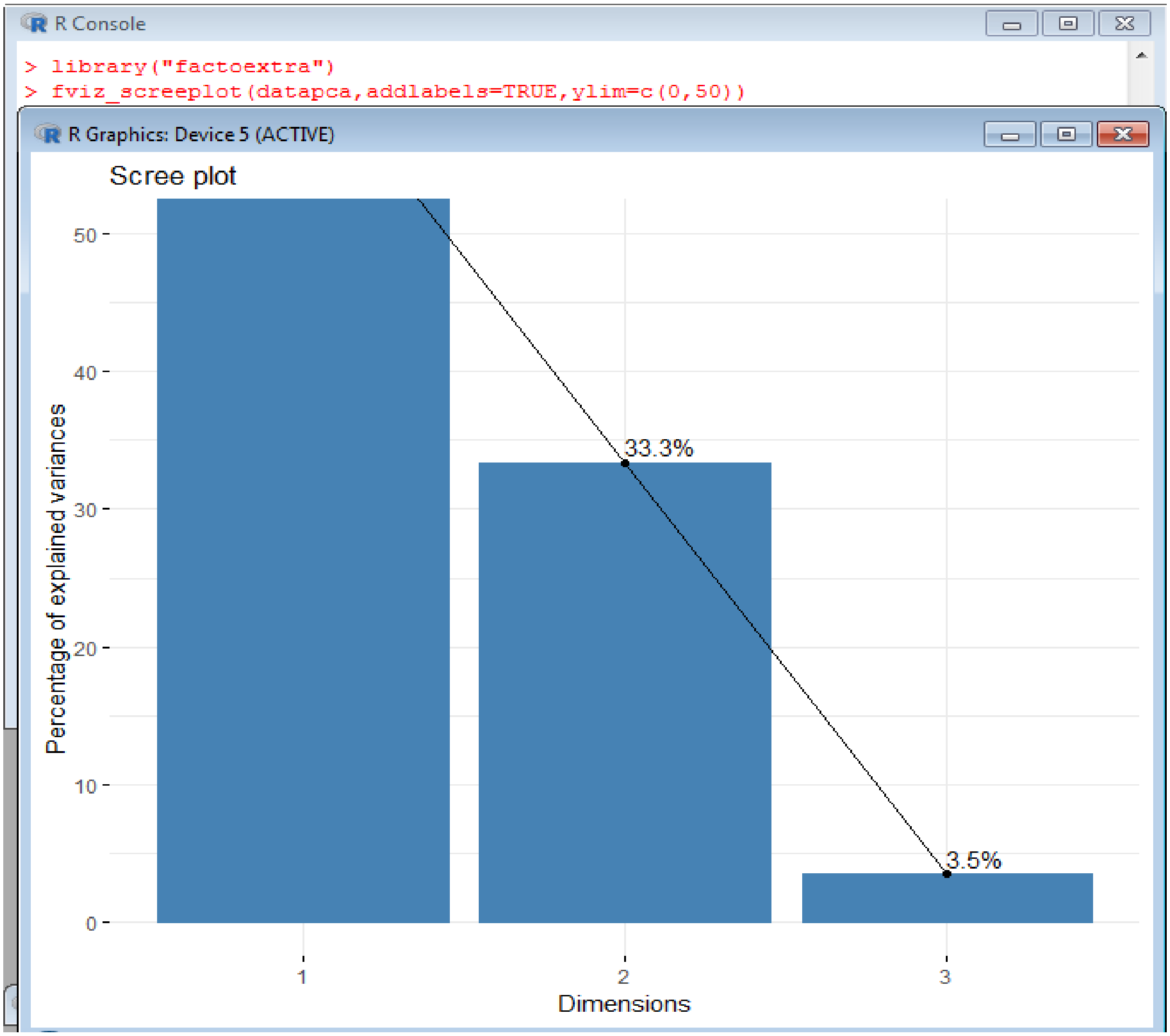
From the above results it is inferred that Component1 orMaths contribute to the maximum variance i.e. 63.21% and together with Component 2 which is Art they can achieve 96.54% variance of the target variable.

R Console

```
> datapca$var
$`coord`
            Dim.1      Dim.2       Dim.3
Math    0.9737611  0.0000000 -0.22757256
English 0.9180708 -0.3333333  0.21455747
Art     0.3245870  0.9428090  0.07585752

$cor
            Dim.1      Dim.2       Dim.3
Math    0.9737611  0.0000000 -0.22757256
English 0.9180708 -0.3333333  0.21455747
Art     0.3245870  0.9428090  0.07585752

$cos2
            Dim.1     Dim.2       Dim.3
Math    0.9482107 0.0000000 0.051789271
English 0.8428540 0.1111111 0.046034908
Art     0.1053567 0.8888889 0.005754363

$contrib
            Dim.1    Dim.2     Dim.3
Math    50.000000  0.00000 50.000000
English 44.444444 11.11111 44.444444
Art      5.555556 88.88889  5.555556
```

R Console

```
> datapca$var$coord
            Dim.1      Dim.2       Dim.3
Math    0.9737611  0.0000000 -0.22757256
English 0.9180708 -0.3333333  0.21455747
Art     0.3245870  0.9428090  0.07585752
```



Conclusion: From the Scree plot and the eigen values table we can conclude that the feature "Maths" plays the most important role in prediction of score and the second principal component is Art as both of them account for the highest variance.

Case study 2

Iris record:

Iris contain the features of a flower having 3 families. The dataset consists of 150 records.

R Console

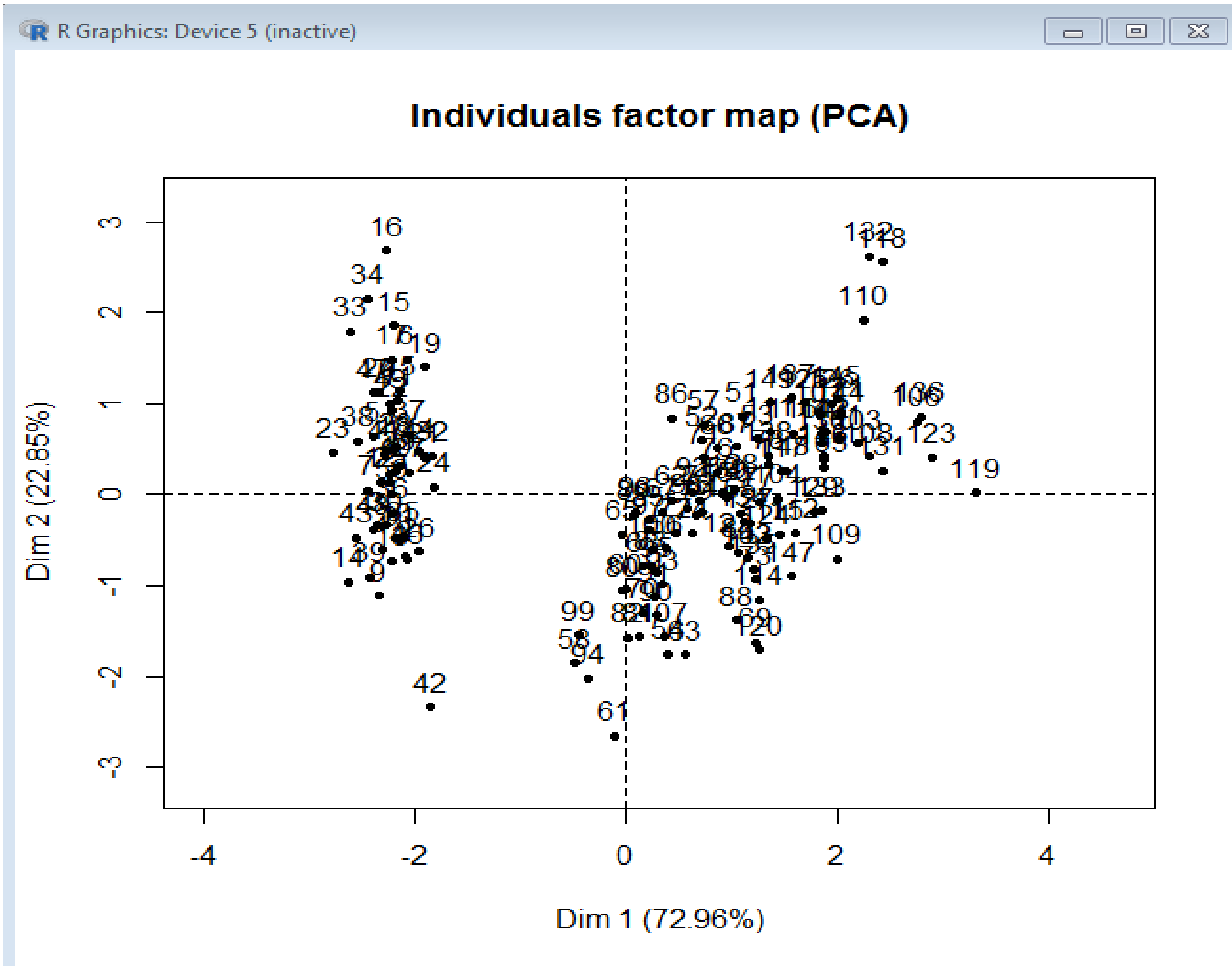
```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> x=iris[,-5]
> x
    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
```

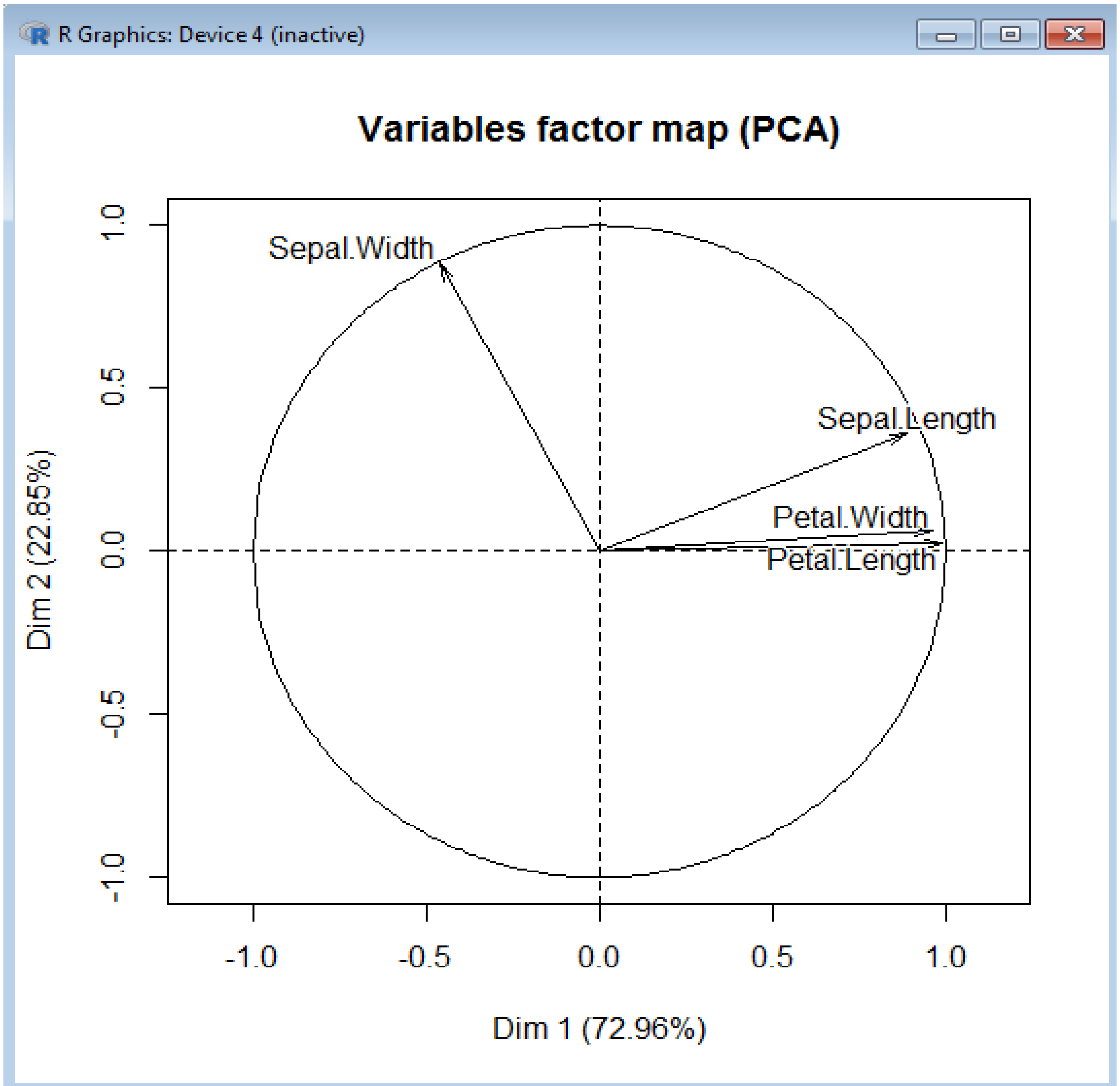
R Console

```
> cov_iris=cov(x)
> cov_iris
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

R Console

```
> irispca=PCA(x,ncp=3,graph=TRUE)
> irispca
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 150 individuals, described by 4 variables
*The results are available in the following objects:

   name               description
1  "$eig"             "eigenvalues"
2  "$var"             "results for the variables"
3  "$var$coord"       "coord. for the variables"
4  "$var$cor"         "correlations variables - dimensions"
5  "$var$cos2"        "cos2 for the variables"
6  "$var$contrib"     "contributions of the variables"
7  "$ind"             "results for the individuals"
8  "$ind$coord"       "coord. for the individuals"
9  "$ind$cos2"        "cos2 for the individuals"
10 "$ind$contrib"     "contributions of the individuals"
11 "$call"            "summary statistics"
12 "$call$centre"     "mean of the variables"
13 "$call$ecart.type" "standard error of the variables"
14 "$call$row.w"      "weights for the individuals"
15 "$call$col.w"      "weights for the variables"
```

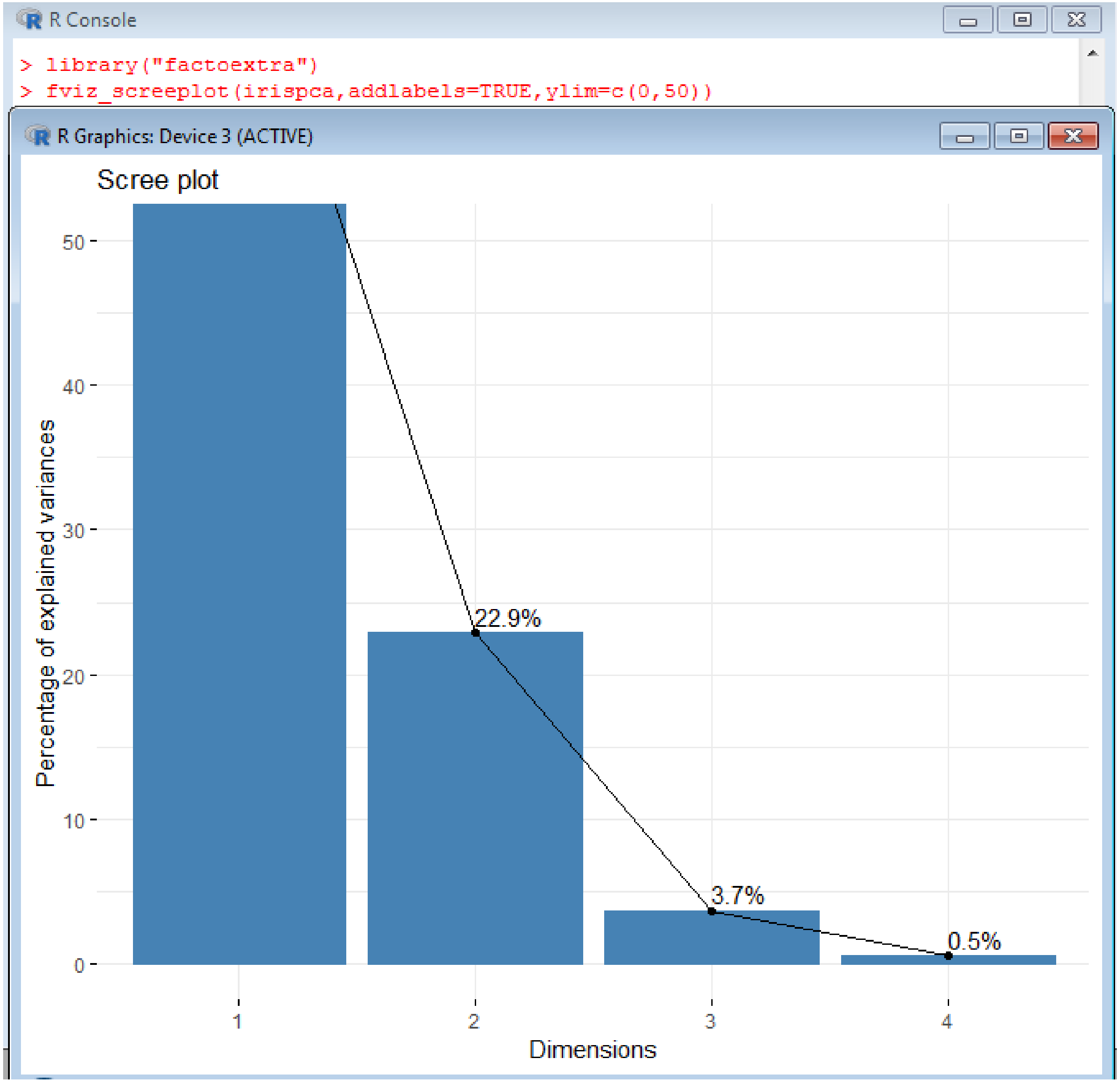

R Graphics: Device 5 (inactive)
Individuals factor map (PCA)
Dim 2 (22.85%)
Dim 1 (72.96%)
-4
-2
0
2
4
3
2
1
0
-1
-2
-3


R Graphics: Device 4 (inactive)
Variables factor map (PCA)
Sepal.Width
Sepal.Length
Petal.Width
Petal.Length
Dim 2 (22.85%)
Dim 1 (72.96%)
-1.0
-0.5
0.0
0.5
1.0

R Console

```
> summary(irispca)

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


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

Individuals (the 10 first)
                  Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
1             |  2.319 | -2.265  1.172  0.954 |  0.480  0.168  0.043 | -0.128
2             |  2.202 | -2.081  0.989  0.893 | -0.674  0.331  0.094 | -0.235
3             |  2.389 | -2.364  1.277  0.979 | -0.342  0.085  0.020 |  0.044
4             |  2.378 | -2.299  1.208  0.935 | -0.597  0.260  0.063 |  0.091
5             |  2.476 | -2.390  1.305  0.932 |  0.647  0.305  0.068 |  0.016
6             |  2.555 | -2.076  0.984  0.660 |  1.489  1.617  0.340 |  0.027
7             |  2.468 | -2.444  1.364  0.981 |  0.048  0.002  0.000 |  0.335
8             |  2.246 | -2.233  1.139  0.988 |  0.223  0.036  0.010 | -0.089
9             |  2.592 | -2.335  1.245  0.812 | -1.115  0.907  0.185 |  0.145
10            |  2.249 | -2.184  1.090  0.943 | -0.469  0.160  0.043 | -0.254
                ctr   cos2
1             0.074  0.003 |
2             0.250  0.011 |
3             0.009  0.000 |
4             0.038  0.001 |
5             0.001  0.000 |
6             0.003  0.000 |
7             0.511  0.018 |
8             0.036  0.002 |
9             0.096  0.003 |
10            0.293  0.013 |
```

```
Variables
                Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
Sepal.Length |  0.890 27.151  0.792 |  0.361 14.244  0.130 | -0.276 51.778
Sepal.Width  | -0.460  7.255  0.212 |  0.883 85.247  0.779 |  0.094  5.972
Petal.Length |  0.992 33.688  0.983 |  0.023  0.060  0.001 |  0.054  2.020
Petal.Width  |  0.965 31.906  0.931 |  0.064  0.448  0.004 |  0.243 40.230
               cos2
Sepal.Length  0.076 |
Sepal.Width   0.009 |
Petal.Length  0.003 |
Petal.Width   0.059 |
```



Conclusion: Sepal.Width has the maximum contribution to Component 2 and Petal.Length and Petal.Width has the maximum contribution to component 1

Assessment No. 04

AIM: Practical of Clustering
K Means Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data.In k means clustering, we have the specify the number of clusters we want the data to be grouped into. The algorithm randomly assigns each observation to a cluster, and finds the centroid of each cluster. Then, the algorithm iterates through two steps:

1. Reassign data points to the cluster whose centroid is closest.
2. Calculate new centroid of each cluster.
3. These two steps are repeated till the within cluster variation cannot be reduced any further.
4. The within cluster variation is calculated as the sum of the Euclidean distance between the data points and their respective cluster centroids.

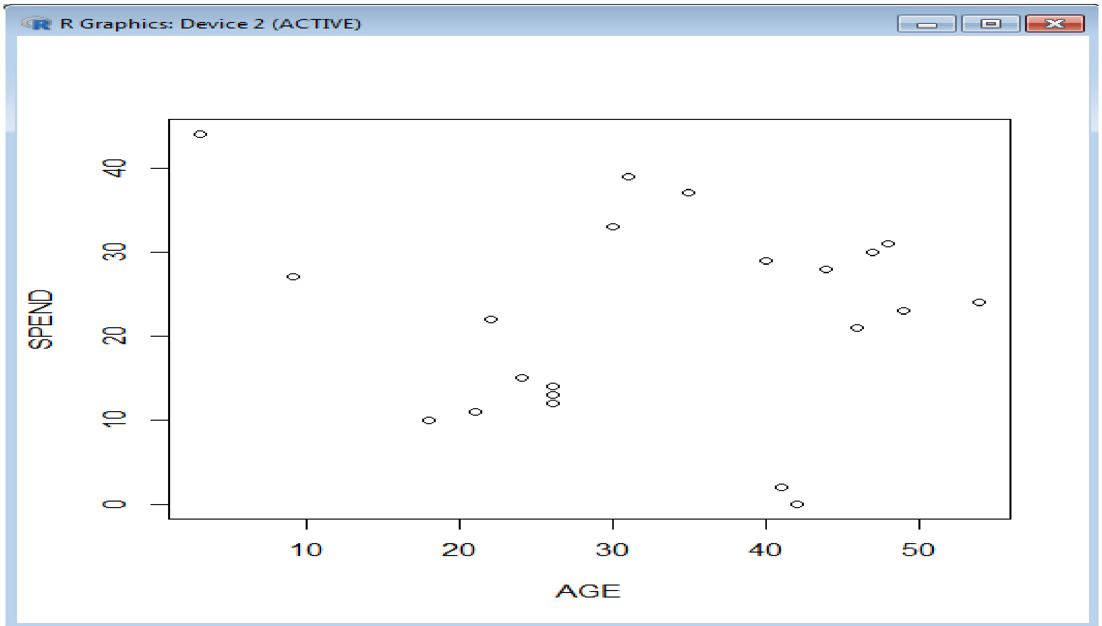
Case study 1

Data: Dataset consists of a sample containing the age of a person and amount of his or her monthly expenditure in thousand.

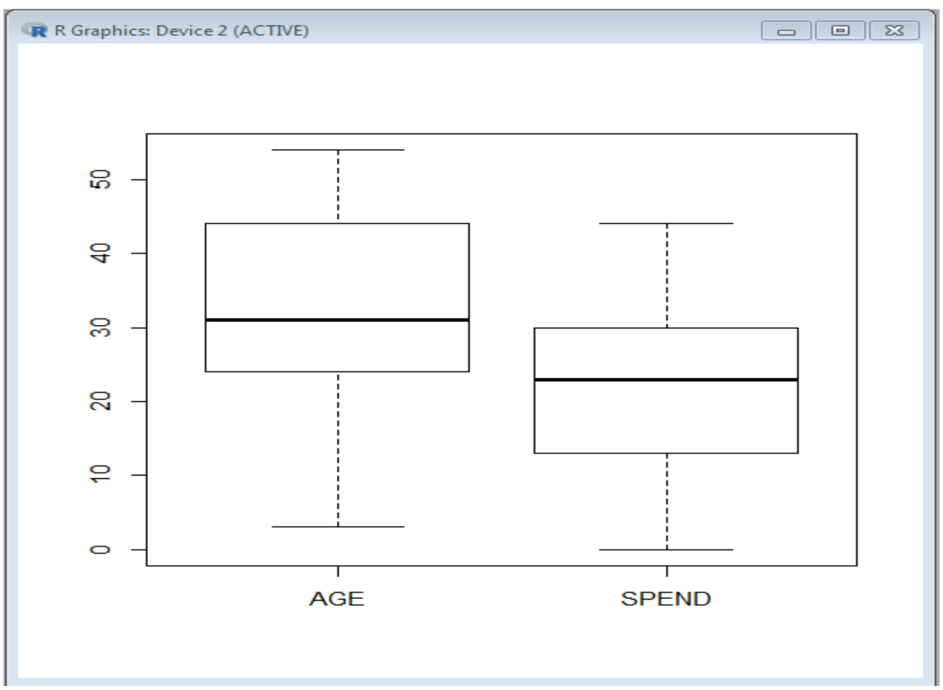
df=read.csv("C:/TYCS A-11/AGE.csv")

```
R Console
> df=read.csv("C:/TYCS A-11/AGE.csv")
> df
   AGE SPEND
1   18    10
2   21    11
3   22    22
4   24    15
5   26    12
6   26    13
7   26    14
8   30    33
9   31    39
10  35    37
11   3    44
12   9    27
13  40    29
14  41     2
15  42     0
16  44    28
17  46    21
18  47    30
19  48    31
20  49    23
21  54    24
> library()
```

#plot(df)



#boxplot(df)



Make the cluster

>set.seed(20)

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

> c1

```
> set.seed(20)
> c1=kmeans(df[,1:2],3)
> c1
K-means clustering with 3 clusters of sizes 8, 5, 8

Cluster means:
     AGE  SPEND
1 45.375 27.875
2 19.000 33.000
3 28.000  9.625

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

Within cluster sum of squares by cluster:
[1] 420.750 944.000 759.875
 (between_SS / total_SS =  68.0 %)

Available components:

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

Case study 2

#SHOW THE IRIS DATA SET

Data: Dataset contains the features of iris flower and their corresponding family.

>iris

R Console

```
> 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
```

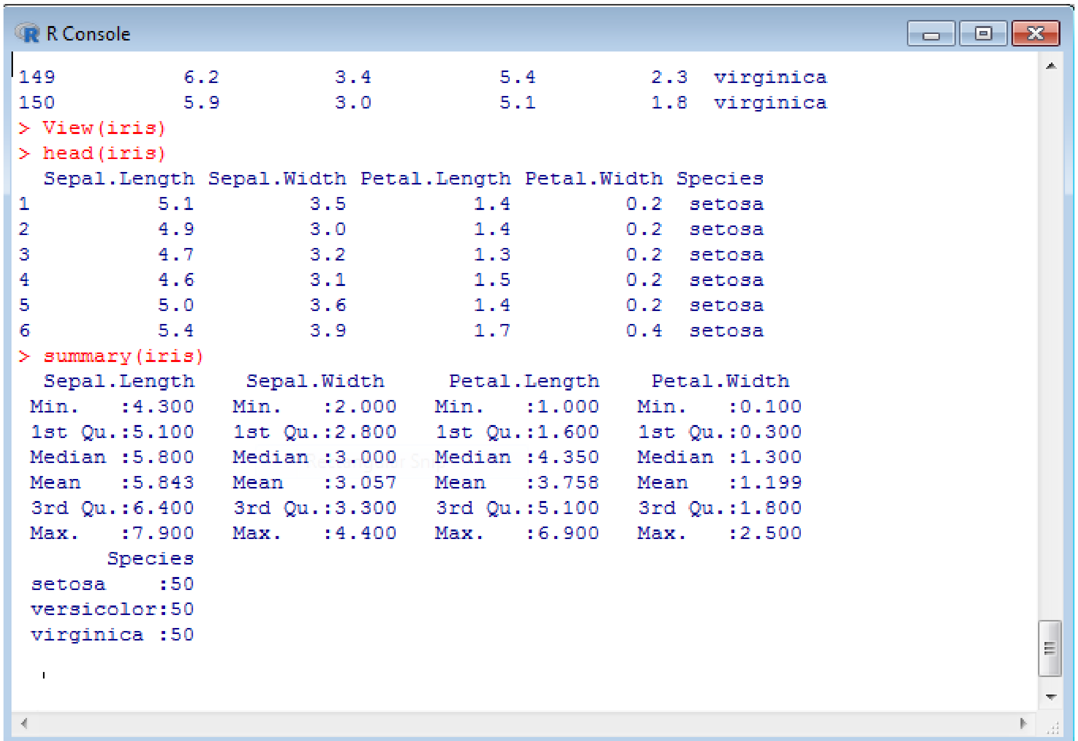
##View(iris) in a tabular format

Data: 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

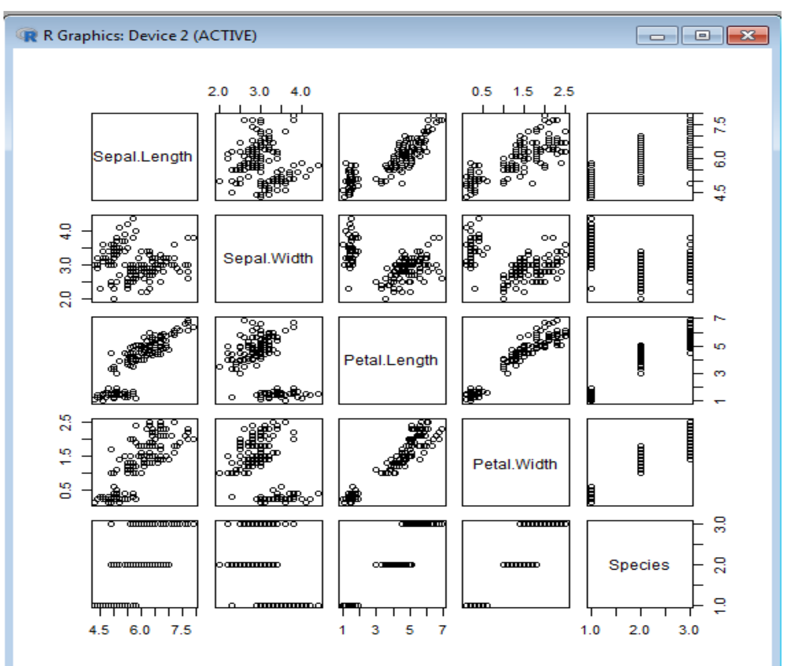
#To print the first 6 records along with the column headers of Iris

>head(iris)

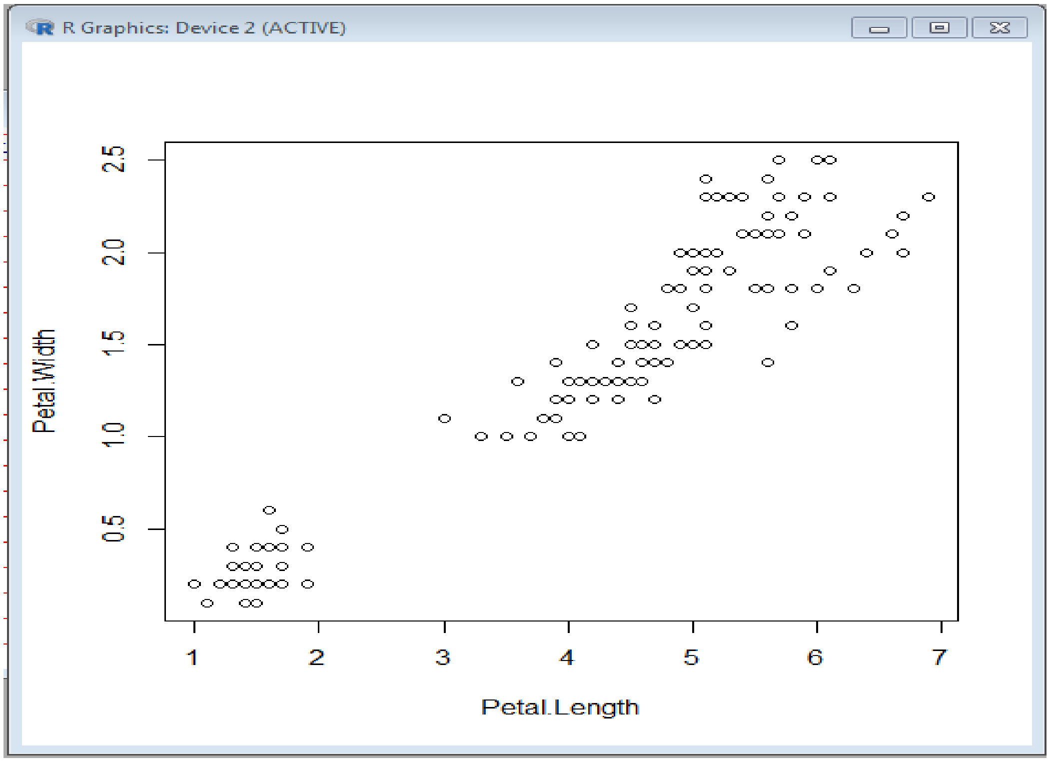


```
R Console
149          6.2          3.4           5.4          2.3  virginica
150          5.9          3.0           5.1          1.8  virginica
> View(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 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
       Species
 setosa    :50
 versicolor:50
 virginica :50
```

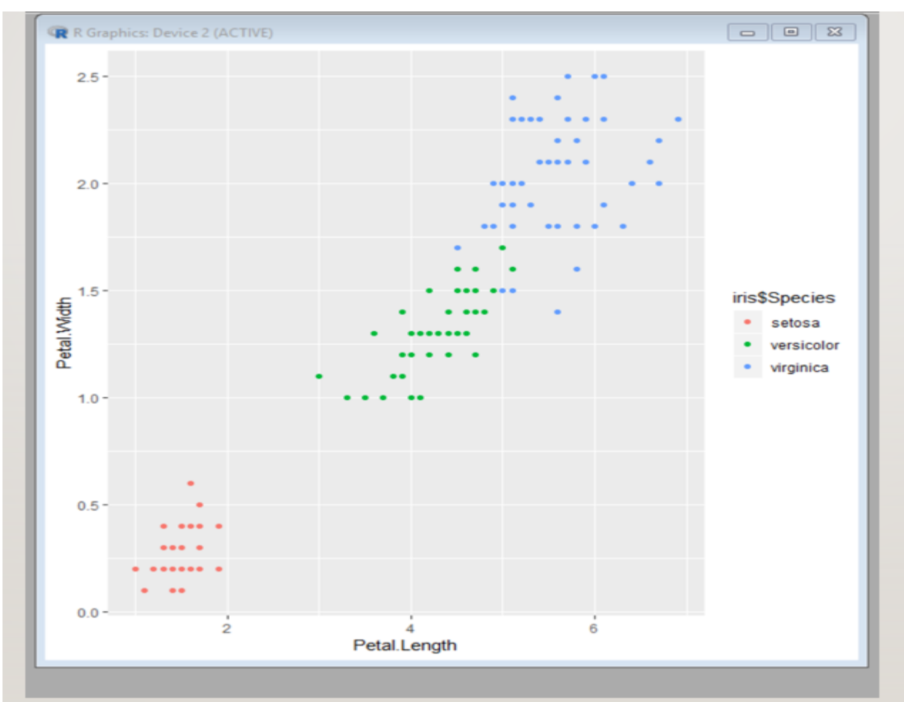
>plot(iris)



```
>plot(iris[,3:4])
```




```
>library(ggplot2)
>ggplot(iris,aes(Petal.Length,Petal.Width,col=iris$Species))+geom_point()
```



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

> kmeansc1

```
R Console
> kmeansc1=kmeans(iris[,3:4],3)
> kmeansc1
K-means clustering with 3 clusters of sizes 50, 46, 54

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

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 2 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 2 2 2 2
[112] 2 2 2 2 2 2 2 2 3 2 2 2 3 2 2 3 3 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2
[149] 2 2

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

Available components:

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

The k-means algorithm takes as input the number of clusters to generate, k, and a set of observation vectors to cluster. It returns a set of centroids, one for each of the k clusters. An observation vector is classified with the cluster number or centroid index of the centroid closest to it.The clustering vectors contain 150 entries for each flower which indicates the cluster to which it belongs.

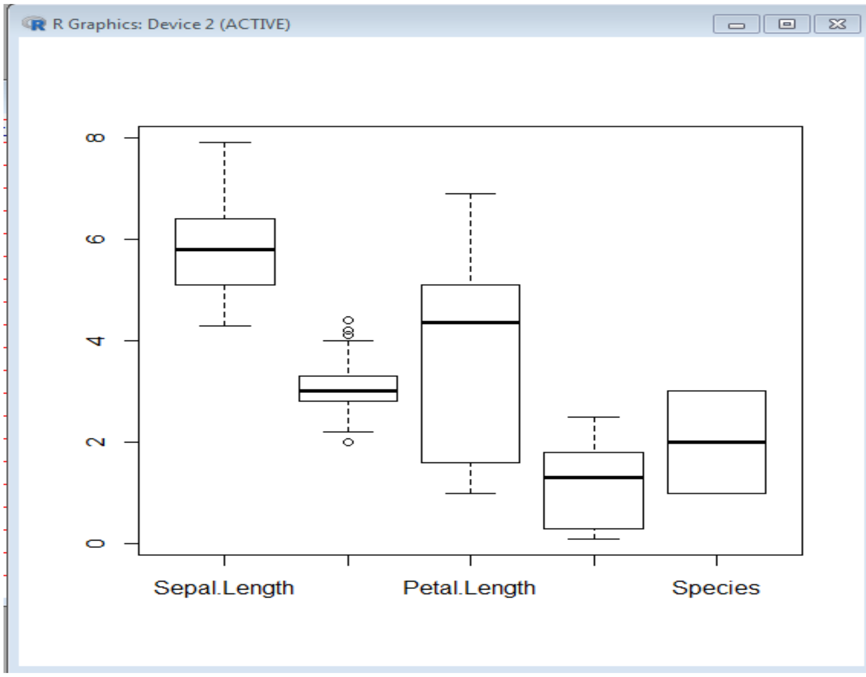
PRINT CONFUSION MATRIX

>table(kmeansc1$cluster,iris$Species)

```
> table(kmeansc1$cluster,iris$Species)

    setosa versicolor virginica
  1     50          0         0
  2      0          2        44
  3      0         48         6
> 
```

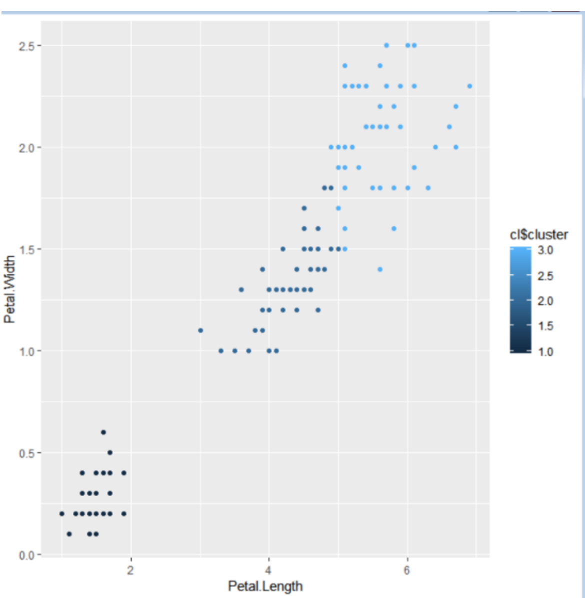
CALCULATION OF ACCURACY 94.6%

>boxplot(iris)



#plotting the cluster

>ggplot(iris,aes(Petal.Length,Petal.Width,col=cl$cluster))+geom_point()



CONCLUSION: Thus we have implemented Clustering successfully.

Assessment No.05

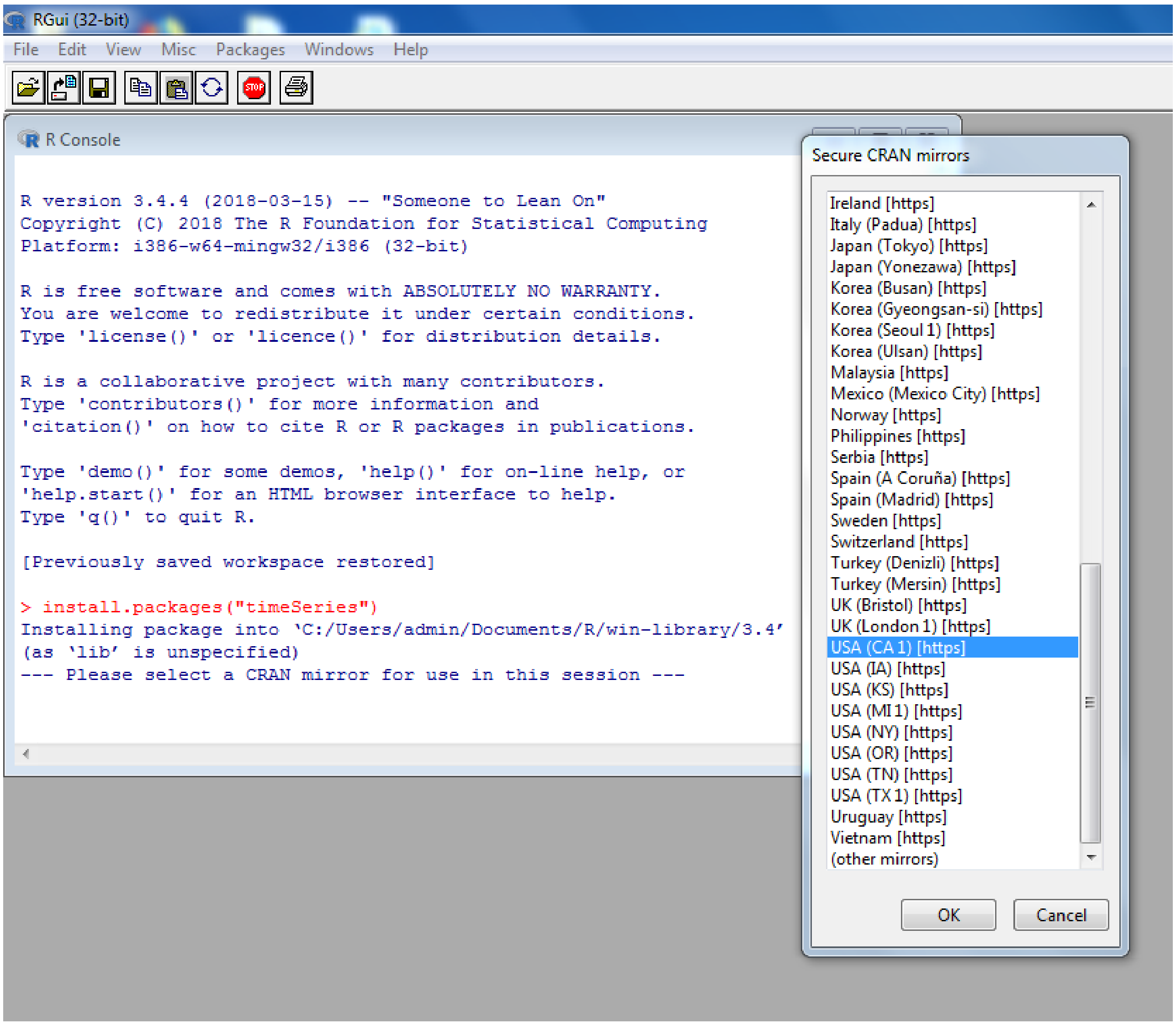
AIM: Practical of Time-series forecasting

Regression and Trend Analysis in Time-Series Data Regression analysis of time-series data has been studied substantially in the fields of statistics and signal analysis. However, one may often need to go beyond pure regression.

1. Trend analysis builds an integrated model using the following four major components or movements to characterize time-series data: 1. Trend or long-term movements: These indicate the general direction in which a time-series graph is moving over time, for example, using weighted moving average and the least squares methods to find trend curves such as the dashed
2. Cyclic movements: These are the long-term oscillations about a trend line or curve.
3. Seasonal variations: These are nearly identical patterns that a time series appears to follow during corresponding seasons of successive years such as holiday shopping seasons. For effective trend analysis, the data often need to be “deseasonalized” based on a seasonal index computed by autocorrelation.
4. Random movements: These characterize sporadic changes due to chance events such as labour disputes or announced personnel changes within companies.

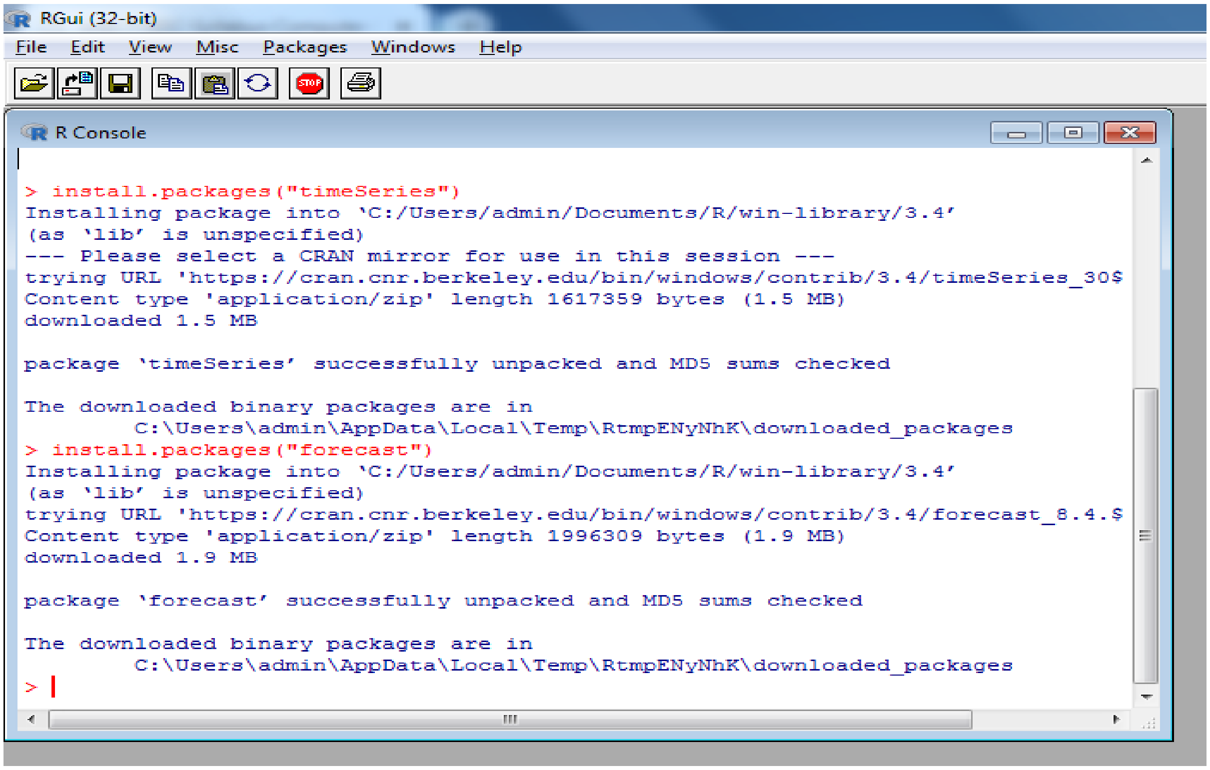
STEP 1: Install timeseries.

#install.packages("timeSeries")



Step 2: Install package forecast

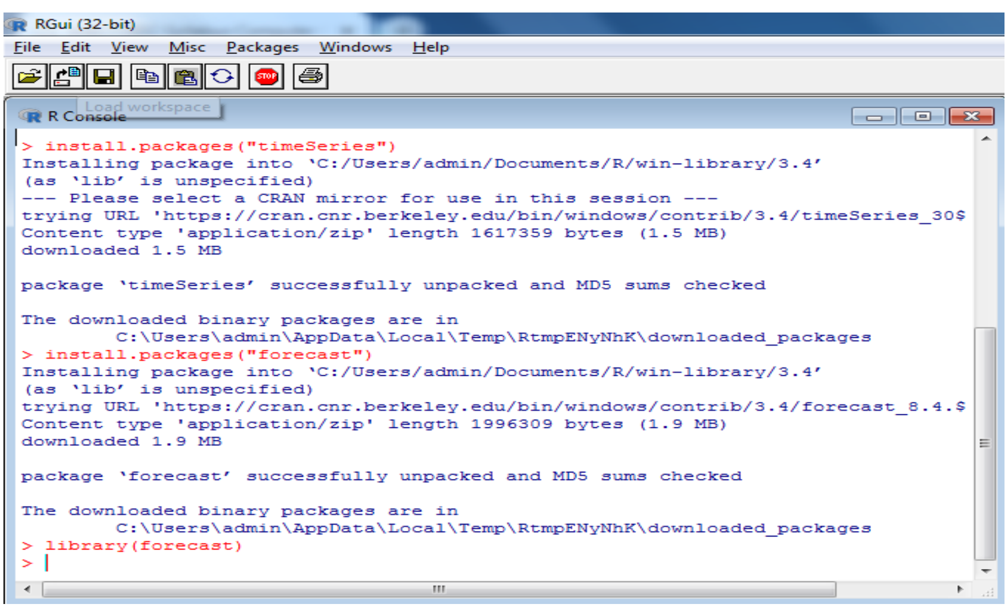
#install.packages(“forecast”)



```
RGui (32-bit)
File  Edit  View  Misc  Packages  Windows  Help
R Console

> install.packages("timeSeries")
Installing package into ‘C:/Users/admin/Documents/R/win-library/3.4’
(as ‘lib’ is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.cnr.berkeley.edu/bin/windows/contrib/3.4/timeSeries_30$
Content type 'application/zip' length 1617359 bytes (1.5 MB)
downloaded 1.5 MB

package ‘timeSeries’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
> install.packages("forecast")
Installing package into ‘C:/Users/admin/Documents/R/win-library/3.4’
(as ‘lib’ is unspecified)
trying URL 'https://cran.cnr.berkeley.edu/bin/windows/contrib/3.4/forecast_8.4.$
Content type 'application/zip' length 1996309 bytes (1.9 MB)
downloaded 1.9 MB

package ‘forecast’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
>
```

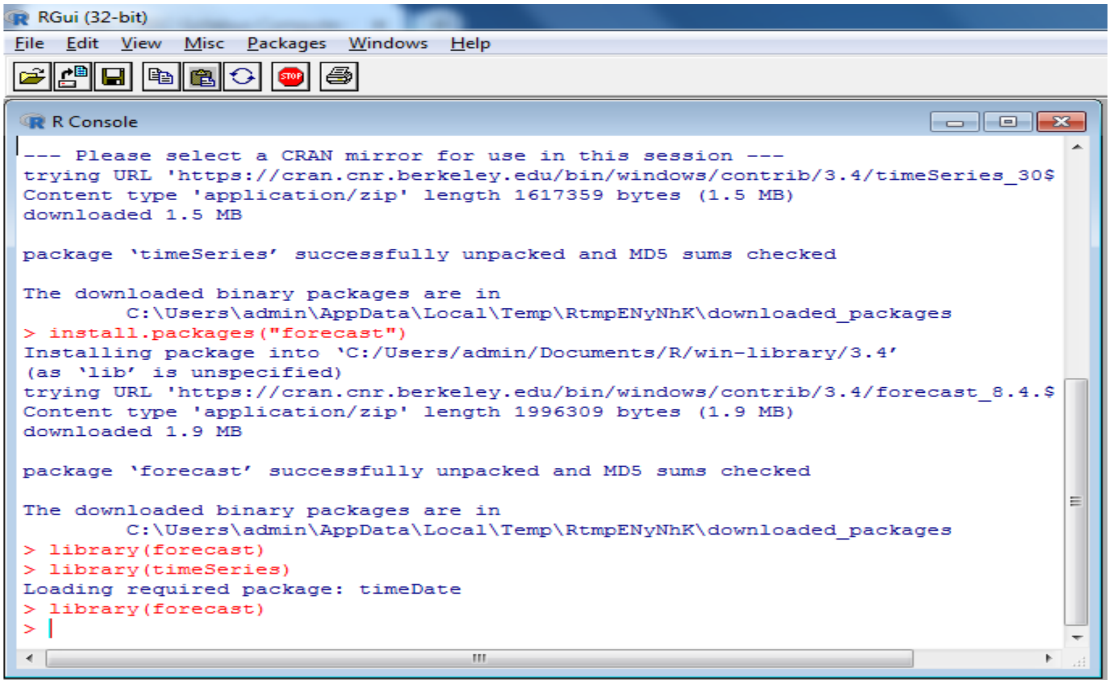
Step 3: library (timeSeries)

#library(forecast)



```
RGui (32-bit)
File  Edit  View  Misc  Packages  Windows  Help
R Console   Load workspace
> install.packages("timeSeries")
Installing package into ‘C:/Users/admin/Documents/R/win-library/3.4’
(as ‘lib’ is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.cnr.berkeley.edu/bin/windows/contrib/3.4/timeSeries_30$
Content type 'application/zip' length 1617359 bytes (1.5 MB)
downloaded 1.5 MB

package ‘timeSeries’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
> install.packages("forecast")
Installing package into ‘C:/Users/admin/Documents/R/win-library/3.4’
(as ‘lib’ is unspecified)
trying URL 'https://cran.cnr.berkeley.edu/bin/windows/contrib/3.4/forecast_8.4.$
Content type 'application/zip' length 1996309 bytes (1.9 MB)
downloaded 1.9 MB

package ‘forecast’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
> library(forecast)
>
```

Step 4: library forecast

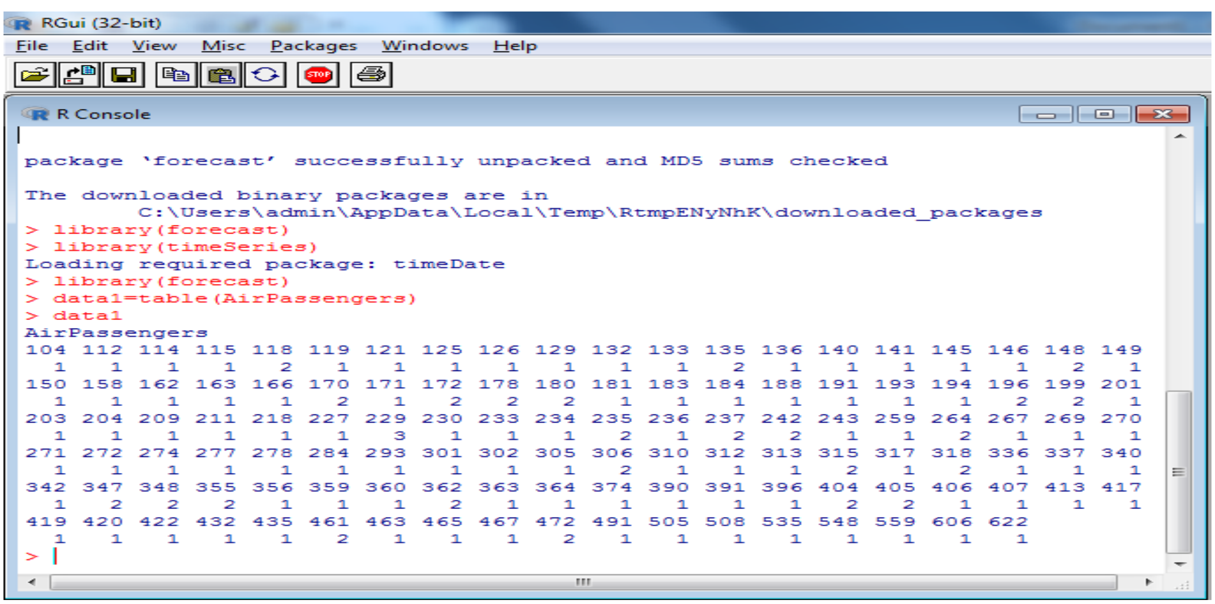



```
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.cnr.berkeley.edu/bin/windows/contrib/3.4/timeSeries_30$
Content type 'application/zip' length 1617359 bytes (1.5 MB)
downloaded 1.5 MB

package 'timeSeries' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
> install.packages("forecast")
Installing package into 'C:/Users/admin/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
trying URL 'https://cran.cnr.berkeley.edu/bin/windows/contrib/3.4/forecast_8.4.$
Content type 'application/zip' length 1996309 bytes (1.9 MB)
downloaded 1.9 MB

package 'forecast' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
> library(forecast)
> library(timeSeries)
Loading required package: timeDate
> library(forecast)
>
```

Step 5: Air Passengers data

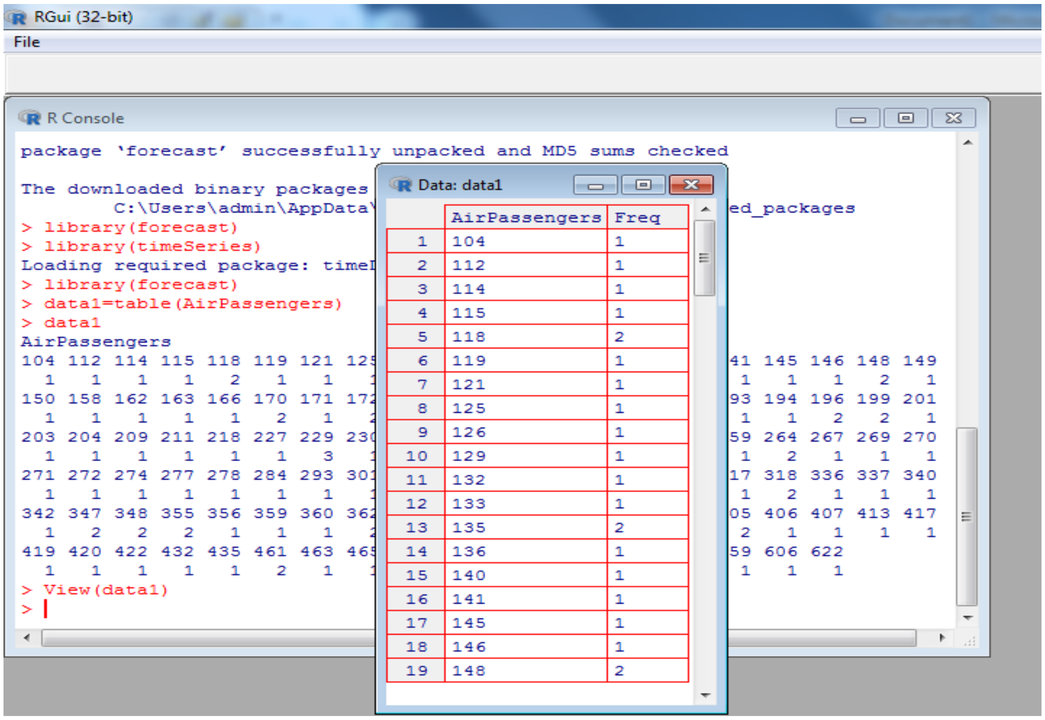
The AirPassenger dataset in R provides monthly totals of a US airline passengers, from 1949 to 1960. This dataset is already of a time series class therefore no further class or date manipulation is required.

#data1=table(AirPassengers)

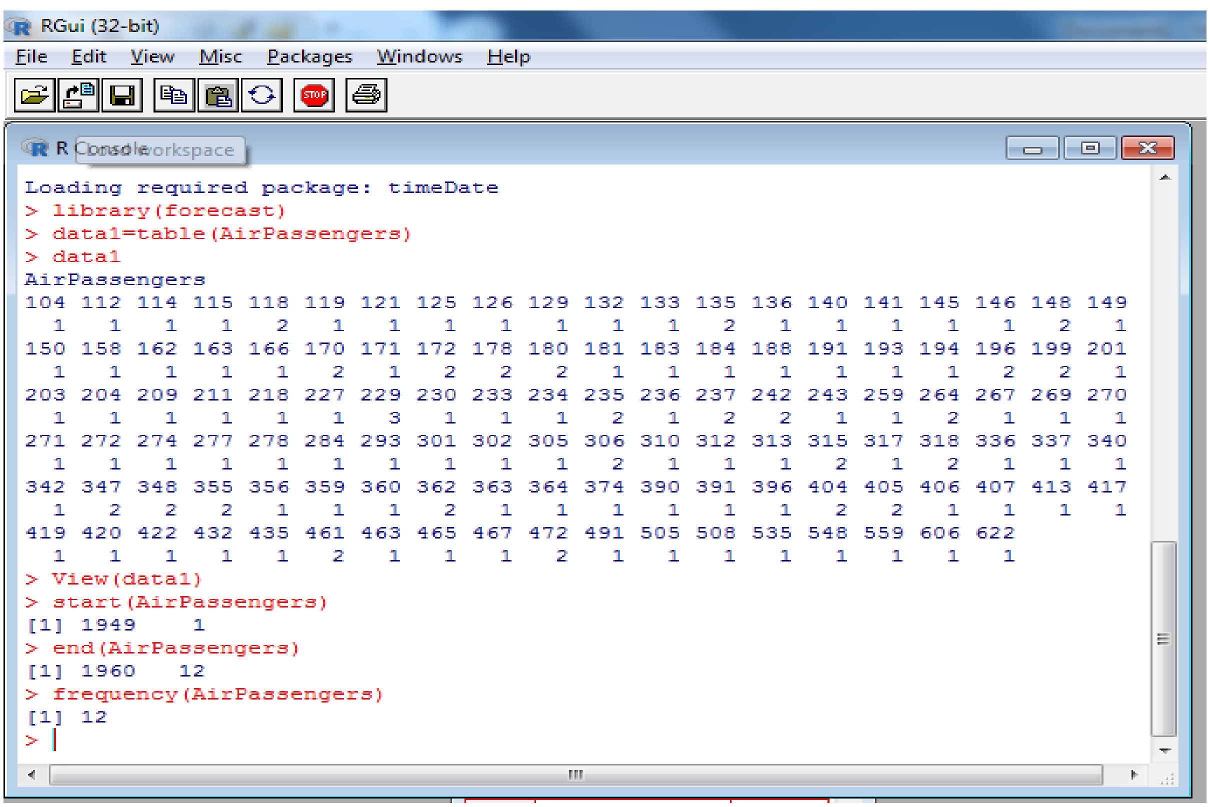



```
package 'forecast' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpENyNhK\downloaded_packages
> library(forecast)
> library(timeSeries)
Loading required package: timeDate
> library(forecast)
> data1=table(AirPassengers)
> data1
AirPassengers
104 112 114 115 118 119 121 125 126 129 132 133 135 136 140 141 145 146 148 149
  1   1   1   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   2   1
150 158 162 163 166 170 171 172 178 180 181 183 184 188 191 193 194 196 199 201
  1   1   1   1   1   2   1   2   2   2   1   1   1   1   1   1   1   2   2   1
203 204 209 211 218 227 229 230 233 234 235 236 237 242 243 259 264 267 269 270
  1   1   1   1   1   1   3   1   1   1   2   1   2   2   1   1   2   1   1   1
271 272 274 277 278 284 293 301 302 305 306 310 312 313 315 317 318 336 337 340
  1   1   1   1   1   1   1   1   1   1   2   1   1   1   2   1   2   1   1   1
342 347 348 355 356 359 360 362 363 364 374 390 391 396 404 405 406 407 413 417
  1   2   2   2   1   1   1   2   1   1   1   1   1   1   2   2   1   1   1   1
419 420 422 432 435 461 463 465 467 472 491 505 508 535 548 559 606 622
  1   1   1   1   1   2   1   1   1   2   1   1   1   1   1   1   1   1
>
```

#View (data1)

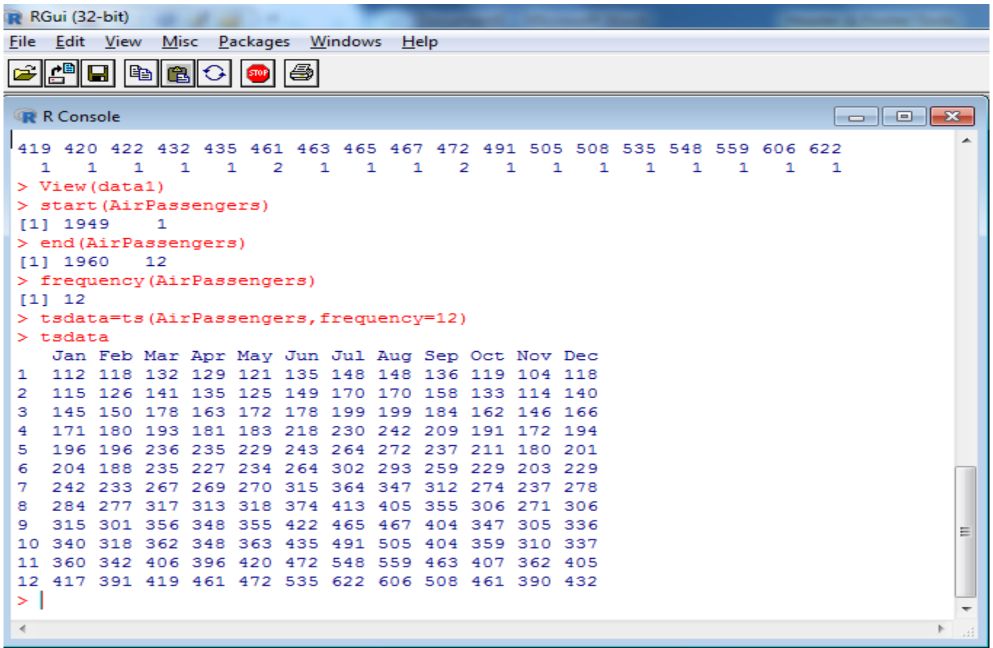


#frequency (AirPassengers)

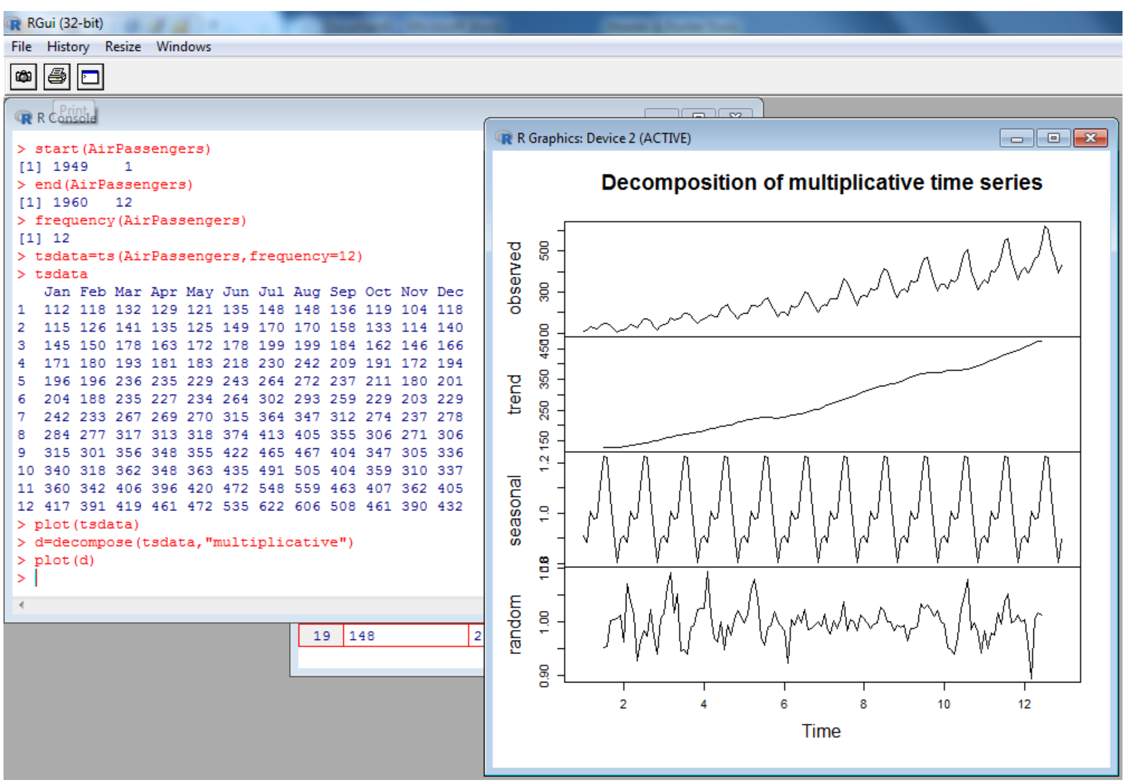


#tsdata=ts(AirPassengers,frequency=12)

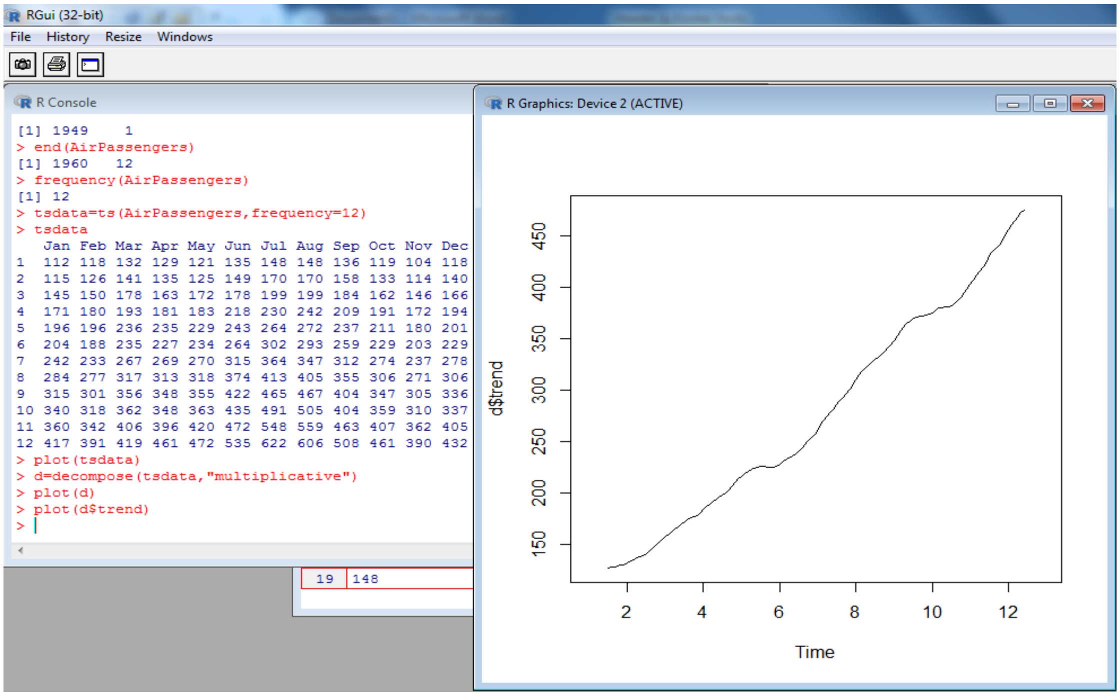
#tsdata

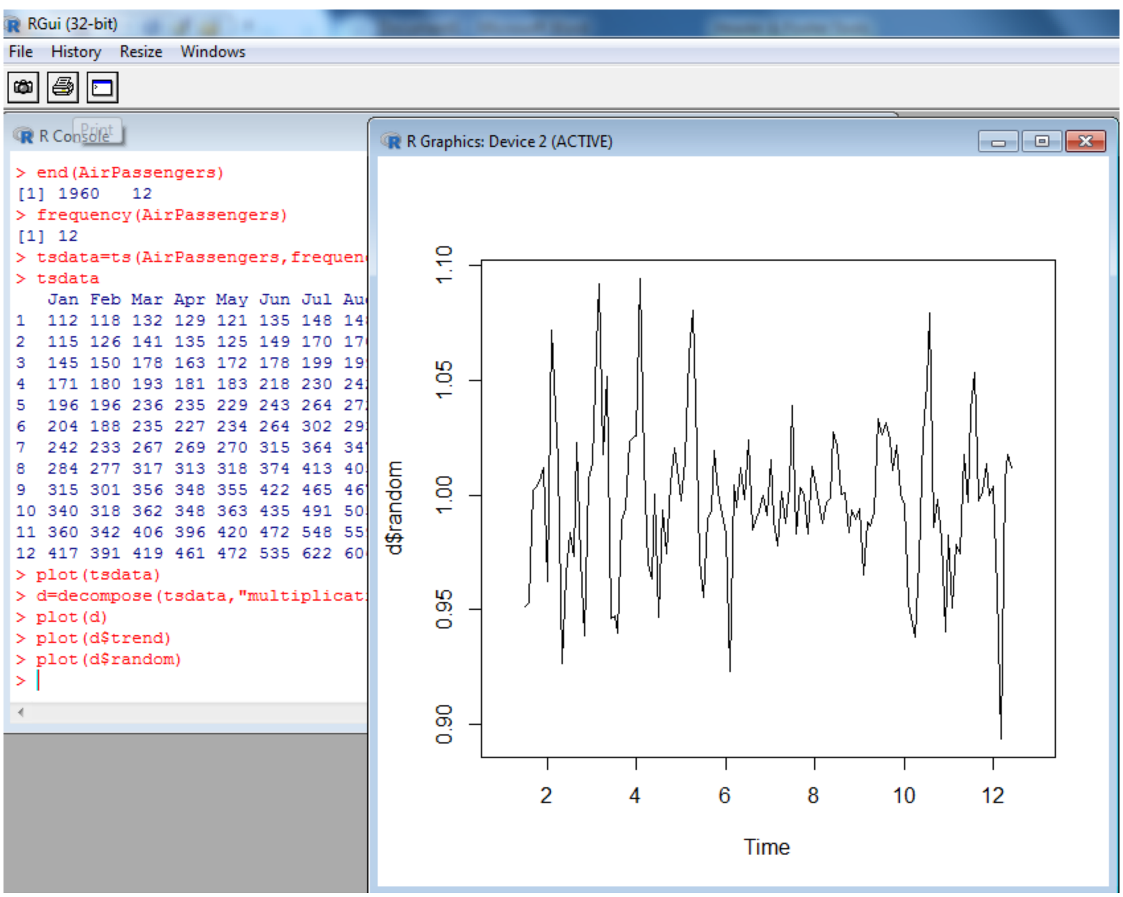


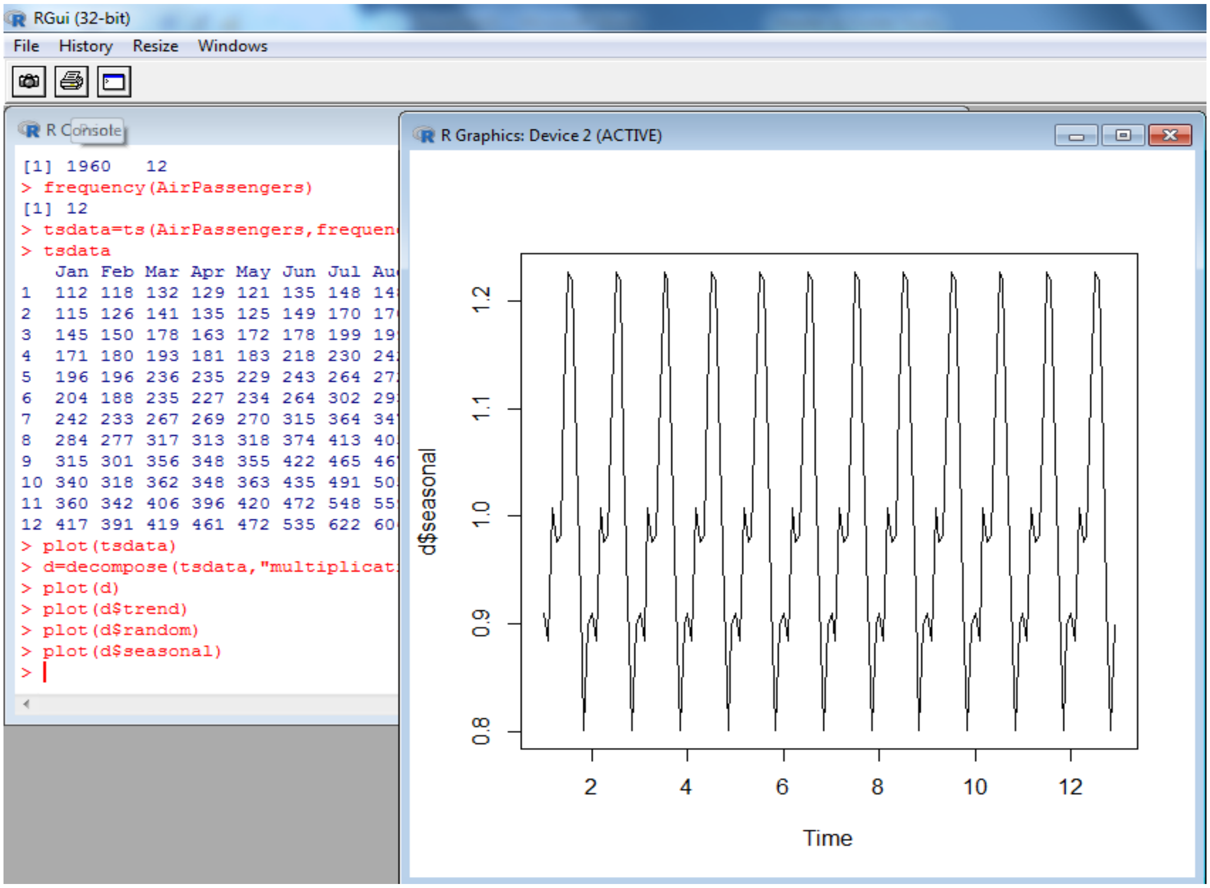
Ploting dataset into random, session, trend, and observed from.

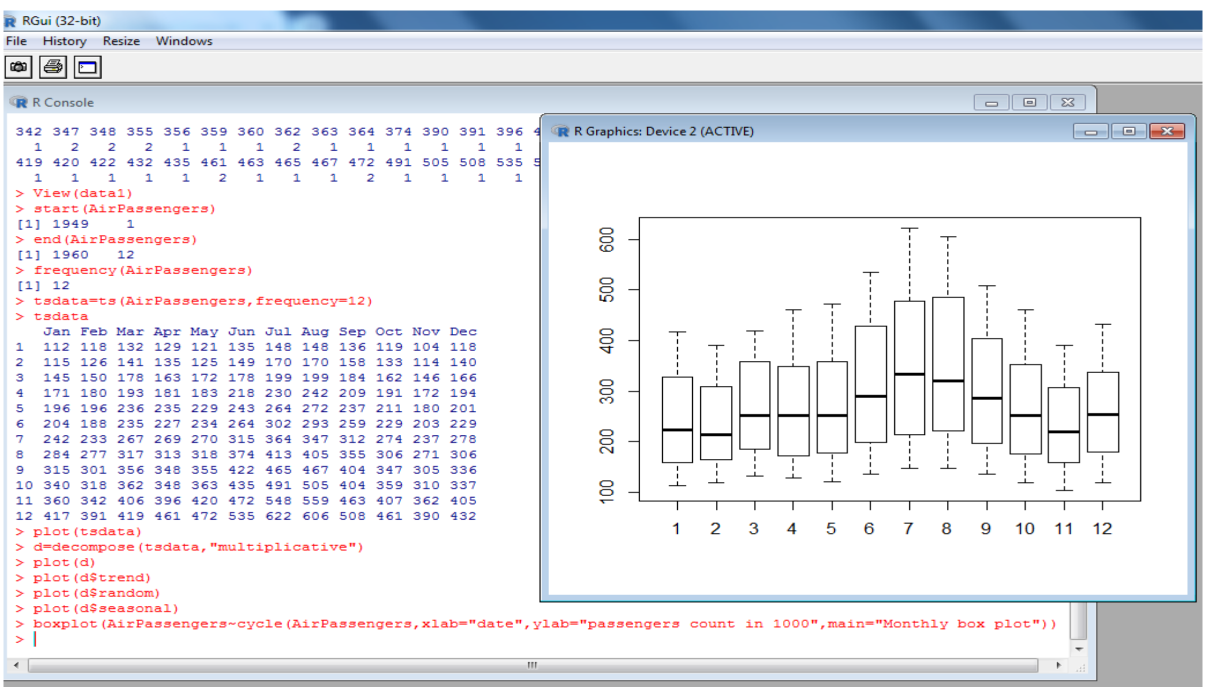


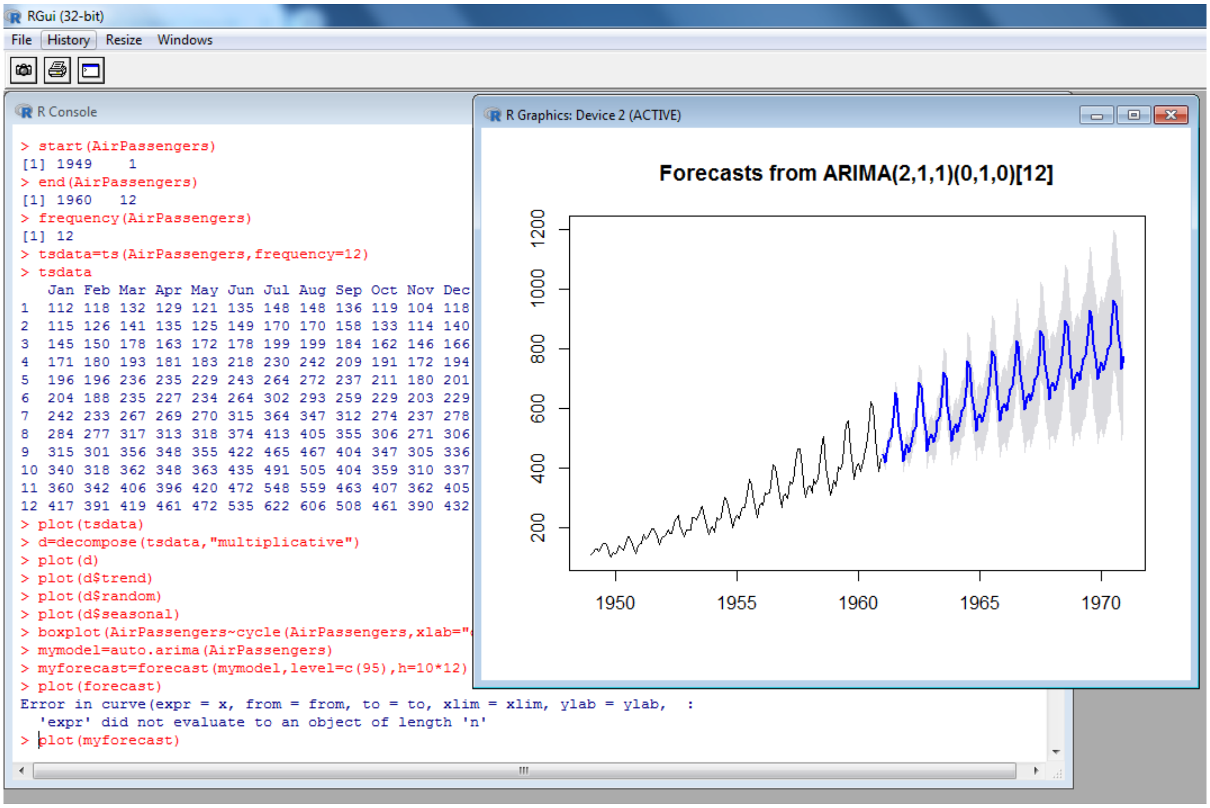
*trend form of data



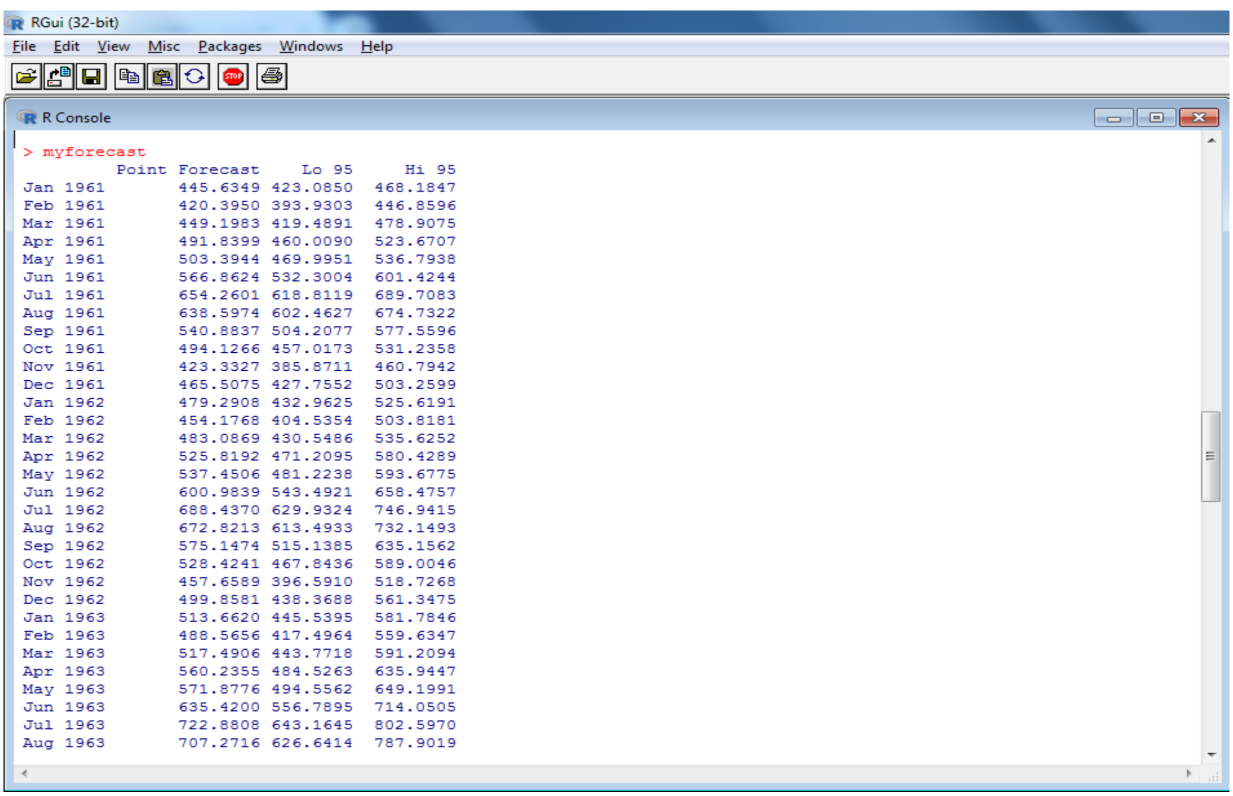



RGui (32-bit)
File History Resize Windows
R Console
[1] 1960 12
> frequency(AirPassengers)
[1] 12
> tsdata=ts(AirPassengers,frequen
> tsdata
> plot(tsdata)
> d=decompose(tsdata,"multiplicat
> plot(d)
> plot(d$trend)
> plot(d$random)
> plot(d$seasonal)
R Graphics: Device 2 (ACTIVE)
d$seasonal
Time

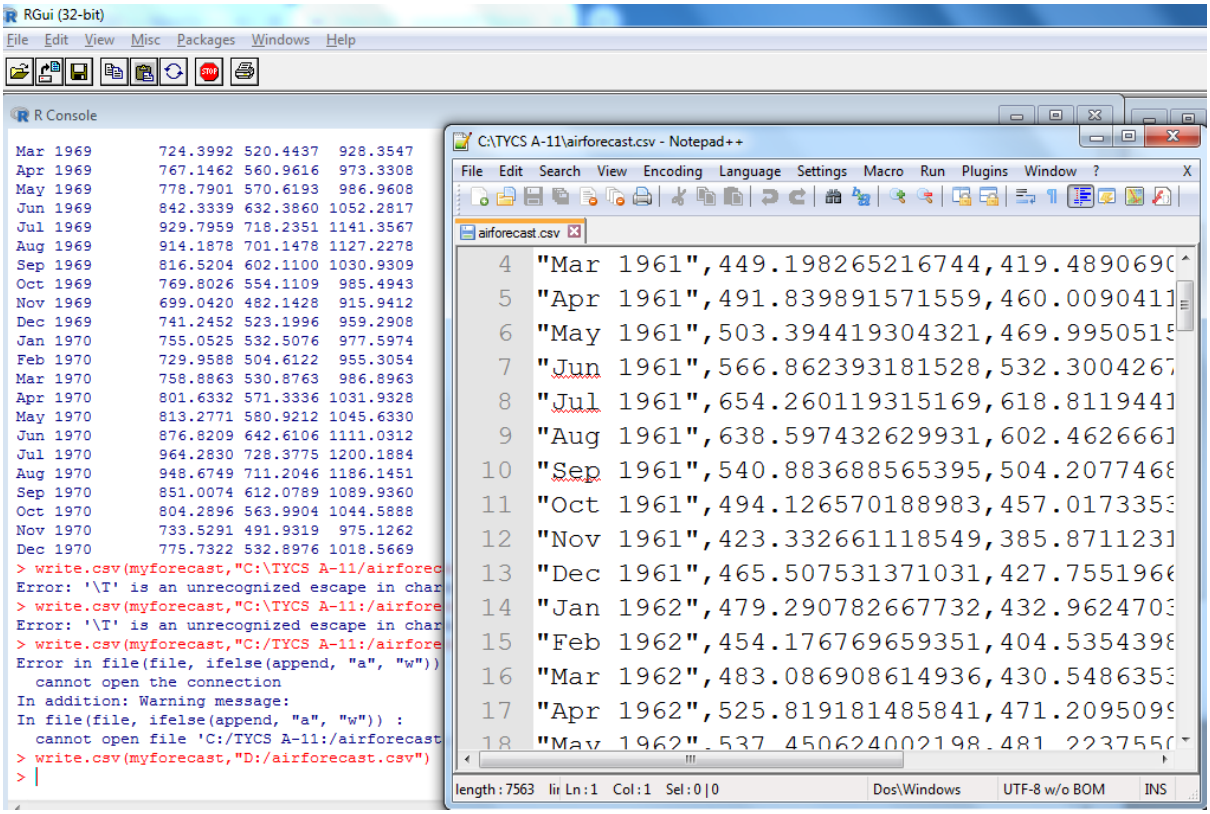

RGui (32-bit)
File History Resize Windows
R Console
> View(data1)
> start(AirPassengers)
[1] 1949 1
> end(AirPassengers)
[1] 1960 12
> frequency(AirPassengers)
[1] 12
> tsdata=ts(AirPassengers,frequency=12)
> tsdata
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1 112 118 132 129 121 135 148 148 136 119 104 118
2 115 126 141 135 125 149 170 170 158 133 114 140
3 145 150 178 163 172 178 199 199 184 162 146 166
4 171 180 193 181 183 218 230 242 209 191 172 194
5 196 196 236 235 229 243 264 272 237 211 180 201
6 204 188 235 227 234 264 302 293 259 229 203 229
7 242 233 267 269 270 315 364 347 312 274 237 278
8 284 277 317 313 318 374 413 405 355 306 271 306
9 315 301 356 348 355 422 465 467 404 347 305 336
10 340 318 362 348 363 435 491 505 404 359 310 337
11 360 342 406 396 420 472 548 559 463 407 362 405
12 417 391 419 461 472 535 622 606 508 461 390 432
> plot(tsdata)
> d=decompose(tsdata,"multiplicative")
> plot(d)
> plot(d$trend)
> plot(d$random)
> plot(d$seasonal)
> boxplot(AirPassengers~cycle(AirPassengers,xlab="date",ylab="passengers count in 1000",main="Monthly box plot"))
R Graphics: Device 2 (ACTIVE)



```
> start(AirPassengers)
[1] 1949    1
> end(AirPassengers)
[1] 1960   12
> frequency(AirPassengers)
[1] 12
> tsdata=ts(AirPassengers,frequency=12)
> tsdata
   Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1  112 118 132 129 121 135 148 148 136 119 104 118
2  115 126 141 135 125 149 170 170 158 133 114 140
3  145 150 178 163 172 178 199 199 184 162 146 166
4  171 180 193 181 183 218 230 242 209 191 172 194
5  196 196 236 235 229 243 264 272 237 211 180 201
6  204 188 235 227 234 264 302 293 259 229 203 229
7  242 233 267 269 270 315 364 347 312 274 237 278
8  284 277 317 313 318 374 413 405 355 306 271 306
9  315 301 356 348 355 422 465 467 404 347 305 336
10 340 318 362 348 363 435 491 505 404 359 310 337
11 360 342 406 396 420 472 548 559 463 407 362 405
12 417 391 419 461 472 535 622 606 508 461 390 432
> plot(tsdata)
> d=decompose(tsdata,"multiplicative")
> plot(d)
> plot(d$trend)
> plot(d$random)
> plot(d$seasonal)
> boxplot(AirPassengers~cycle(AirPassengers,xlab="
> mymodel=auto.arima(AirPassengers)
> myforecast=forecast(mymodel,level=c(95),h=10*12)
> plot(forecast)
Error in curve(expr = x, from = from, to = to, xlim = xlim, ylab = ylab,  :
  'expr' did not evaluate to an object of length 'n'
> plot(myforecast)
```

```
> myforecast
         Point Forecast    Lo 95    Hi 95
Jan 1961       445.6349 423.0850 468.1847
Feb 1961       420.3950 393.9303 446.8596
Mar 1961       449.1983 419.4891 478.9075
Apr 1961       491.8399 460.0090 523.6707
May 1961       503.3944 469.9951 536.7938
Jun 1961       566.8624 532.3004 601.4244
Jul 1961       654.2601 618.8119 689.7083
Aug 1961       638.5974 602.4627 674.7322
Sep 1961       540.8837 504.2077 577.5596
Oct 1961       494.1266 457.0173 531.2358
Nov 1961       423.3327 385.8711 460.7942
Dec 1961       465.5075 427.7552 503.2599
Jan 1962       479.2908 432.9625 525.6191
Feb 1962       454.1768 404.5354 503.8181
Mar 1962       483.0869 430.5486 535.6252
Apr 1962       525.8192 471.2095 580.4289
May 1962       537.4506 481.2238 593.6775
Jun 1962       600.9839 543.4921 658.4757
Jul 1962       688.4370 629.9324 746.9415
Aug 1962       672.8213 613.4933 732.1493
Sep 1962       575.1474 515.1385 635.1562
Oct 1962       528.4241 467.8436 589.0046
Nov 1962       457.6589 396.5910 518.7268
Dec 1962       499.8581 438.3688 561.3475
Jan 1963       513.6620 445.5395 581.7846
Feb 1963       488.5656 417.4964 559.6347
Mar 1963       517.4906 443.7718 591.2094
Apr 1963       560.2355 484.5263 635.9447
May 1963       571.8776 494.5562 649.1991
Jun 1963       635.4200 556.7895 714.0505
Jul 1963       722.8808 643.1645 802.5970
Aug 1963       707.2716 626.6414 787.9019
```



CONCLUSION: Thus we have implemented Time Series Forecast successfully.

PRACTICAL 6

Aim: Simple /Multiple Linear Regressions.

Linear regression is a basic and commonly used type of predictive analysis. These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + m*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

#IMPORT DATASET:

Command:

>data=read.csv (“D://tycs/score.csv”)

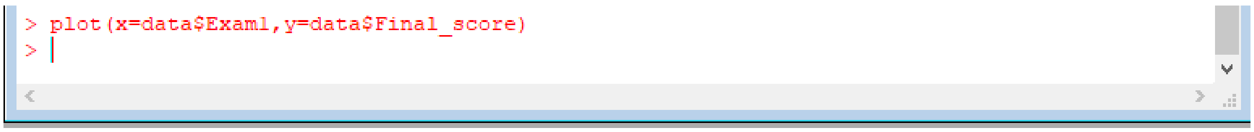
>data

```
R Console
> data= read.csv("D://TYCS21/score.csv")
> data
   Exam1 Exam2 Exam3 Exam4 Final_score Grade
1     60    10    16   7.0       40.79     C
2     90     0     0   0.0       69.23     B
3    130    20    24   1.0       76.75     B
4    130    10    24   8.5       75.66     B
5     90     5    22   9.5       55.48     C
6    100    30    20   3.0       67.11     B
7    105    20    22   8.0       67.98     B
8    120    40    18  16.0       85.09     A
9    120    20    30  18.0       82.46     A
10   130    45    22  10.5       91.01     A
11    90    40    20   7.0       68.86     B
12   130    30    28  10.5       87.06     A
13   100    30    22   6.5       69.52     B
14     0    30    18   0.0       60.00     B
15     0    30    18   0.0       60.00     B
16    80     0    24   3.0       60.11     B
17   105    40    22   6.5       76.10     B
18    10     0     0   8.0       12.16     D
19   130    35    24   0.0       90.00     A
20     0    15    20   7.0       42.86     C
21    40    10    14   6.0       30.70     D
22    90    15    28   8.5       62.06     B
23   110     0    24   9.5       80.62     A
24    65     5    24   1.0       41.67     C
25    55    15    18   0.0       41.90     C
26   100    50    30  11.5       83.99     A
27    95    40    24   8.0       73.25     B
28     0    10    24   0.0       42.50     C
29     0     0    18   0.0       60.00     B
30    65    20    20   0.0       50.00     C
31   110    25    18   6.0       69.74     B
32   130    45    24   8.0       90.79     A
33   120    40    30   9.0       87.28     A
```

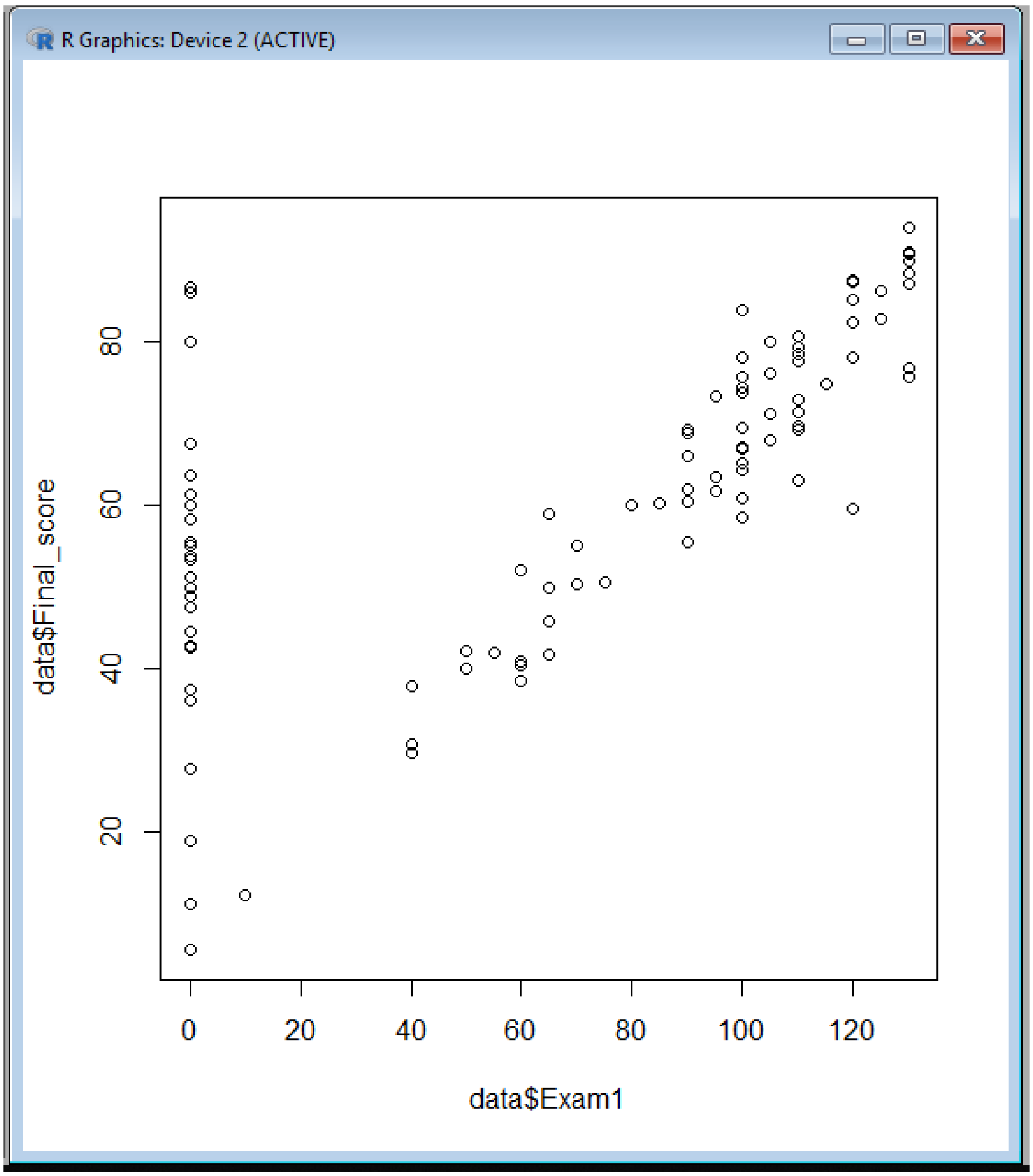
#PLOT THE DATASET:

COMMAND:

>plot(x=data$Exam1,y=data$Final_score)



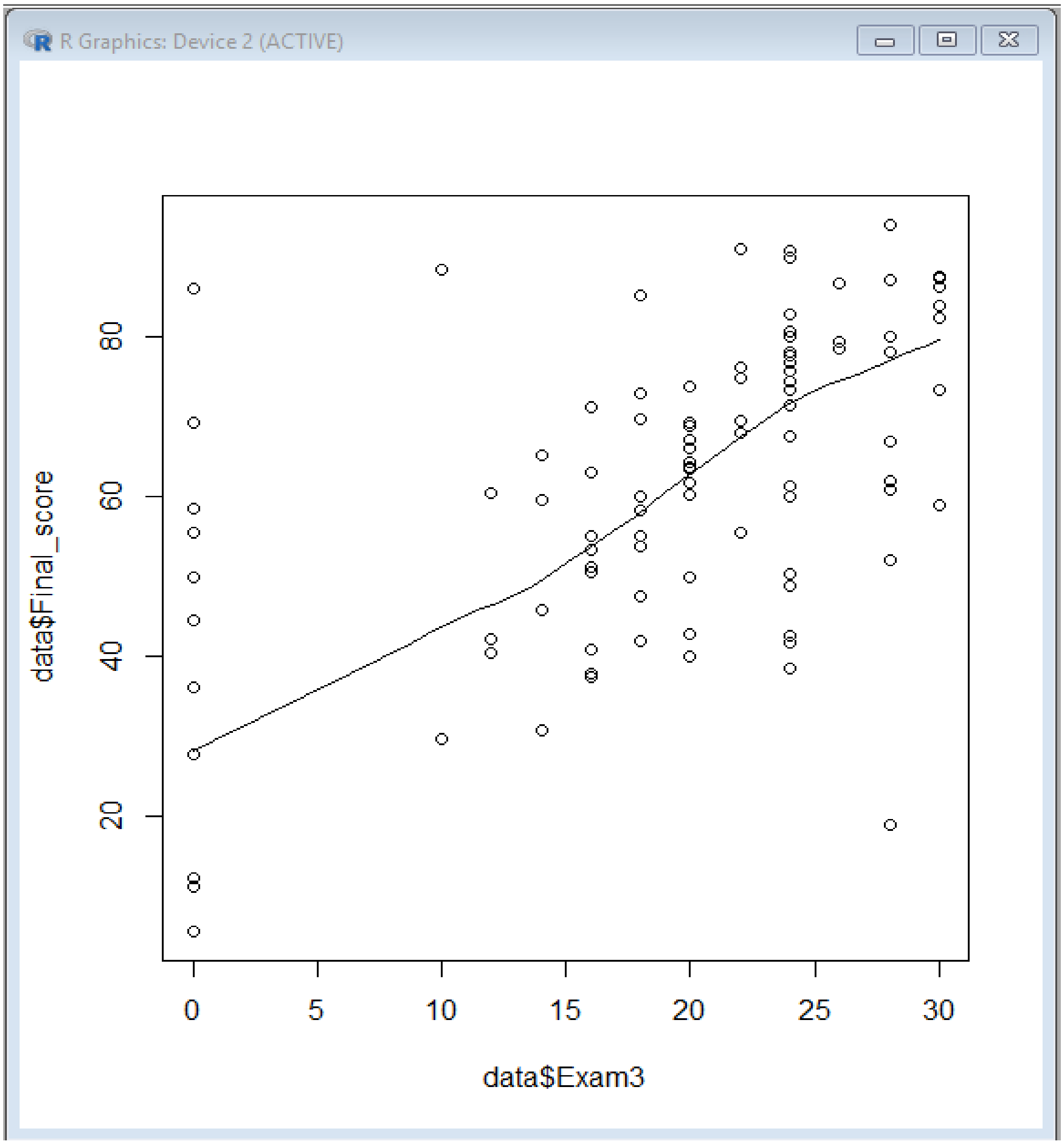
Checking whether the independent variable has a linear relationship with the target variable



#PLOT THE SCATTER DIAGRAM:

>scatter.smooth(x=data$Exam3,y=data$Final_score)

```
> scatter.smooth(x=data$Exam3,y=data$Final_score)
> |
```




```
> cor(data$Exam3,data$Final_score)
[1] 0.6046352
> |
```

#PARTITIONING THE DATABASE INTO TRAINING AND TESTING SET

>s=sample(nrow(data),.7*nrow(data))

>score_tr=data[s,]

>score_test=[-s,]

Score_tr

R Console

```
> s=sample(nrow(data),.7*nrow(data))
> score_tr=data[s,]
> score_test=data[-s,]
> score_tr
    Exam1 Exam2 Exam3 Exam4 Final_score Grade
6     100    30    20   3.0       67.11     B
24     65     5    24   1.0       41.67     C
9     120    20    30  18.0       82.46     A
97      0     0    26   0.0       86.67     A
86    115    20    22   0.0       74.76     B
51      0     0     0  15.5       86.11     A
46      0    25    18   0.0       53.75     C
102    95    20    20   9.5       63.38     B
74     60    15    12   5.0       40.35     C
92      0     0     0   6.5       36.11     D
27     95    40    24   8.0       73.25     B
25     55    15    18   0.0       41.90     C
70     65    15    14  10.5       45.83     C
67      0     0     0   5.0       27.78     D
57    120    40    30   9.5       87.50     A
93     85    30    20   2.5       60.31     B
64    125    30    30  11.5       86.18     A
95      0    25    16   0.0       51.25     C
94      0    30    20  12.5       63.78     B
35    130    45    10  16.5       88.38     A
66     75    15    16   0.0       50.48     C
72    100    15    28   9.5       66.89     B
23    110     0    24   9.5       80.62     A
39      0    25    16   0.0       51.25     C
68      0     0     0  10.0       55.56     C
60    125    25    24   0.0       82.86     A
54      0     0    16   0.0       53.33     C
82      0    25     0   0.0       50.00     C
77      0    15    24   0.0       48.75     C
```

R Console

```
> score_test
    Exam1 Exam2 Exam3 Exam4 Final_score Grade
2      90     0     0   0.0       69.23     B
4     130    10    24   8.5       75.66     B
11     90    40    20   7.0       68.86     B
12    130    30    28  10.5       87.06     A
16     80     0    24   3.0       60.11     B
19    130    35    24   0.0       90.00     A
21     40    10    14   6.0       30.70     D
22     90    15    28   8.5       62.06     B
26    100    50    30  11.5       83.99     A
30     65    20    20   0.0       50.00     C
32    130    45    24   8.0       90.79     A
34     70    20    24   1.0       50.44     C
36      0     0    18  10.0       58.33     C
38     50    30    12   4.0       42.11     C
40     95    20    20   6.0       61.84     B
43      0     0    26   0.0       86.67     A
50    130    40    28  16.5       94.08     A
55    110    25    20   3.0       69.30     B
58    110    35    26  10.0       79.39     B
61    100     0    28   0.0       60.95     B
62      0     0     0   2.0       11.11     D
71      0     0     0   2.0       11.11     D
75     40    20    16  10.5       37.94     D
76    100    35    24   0.0       75.71     B
78    100    15    20   0.0       64.29     B
83    120    20    28  10.0       78.07     B
85      0     0     0   1.0        5.56     D
89      0     0     0   2.0       11.11     D
90      0    30    24   0.0       67.50     B
91    110    25    24   4.0       71.49     B
98    100     0     0   4.0       58.43     C
101   105    30    16  11.5       71.27     B
```

#CREATING A MODEL

```
> linmon=lm(Final_score~Exam3,data=score_tr)
> print(linmod)
Error in print(linmod) : object 'linmod' not found
> print(linmon)

Call:
lm(formula = Final_score ~ Exam3, data = score_tr)

Coefficients:
(Intercept)        Exam3
     39.537        1.119

> |
```

m or (regression coefficient) =1.119

Intercept=39.537

#PREDICTING THE OUTPUT ON TEST DATASET

```
> pdata=predict(linmon,score_test)
> summary(linmon)

Call:
lm(formula = Final_score ~ Exam3, data = score_tr)

Residuals:
    Min      1Q  Median      3Q     Max
-52.005  -9.967   1.666  10.500  46.573

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  39.5367     4.8090   8.221 7.15e-12 ***
Exam3         1.1189     0.2362   4.737 1.10e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.42 on 70 degrees of freedom
Multiple R-squared:  0.2427,    Adjusted R-squared:  0.2319
F-statistic: 22.44 on 1 and 70 DF,  p-value: 1.101e-05

> |
```

Printing Actuals vs Predicted values

```
> actual_predict=data.frame(cbind(actuals=score_test$Final_score,predicteds=pdata))
> actual_predict
   actuals predicteds
2    69.23   39.53669
4    75.66   66.38965
11   68.86   61.91416
12   87.06   70.86515
16   60.11   66.38965
19   90.00   66.38965
21   30.70   55.20092
22   62.06   70.86515
26   83.99   73.10290
30   50.00   61.91416
32   90.79   66.38965
34   50.44   66.38965
36   58.33   59.67641
38   42.11   52.96317
40   61.84   61.91416
43   86.67   68.62740
50   94.08   70.86515
55   69.30   61.91416
58   79.39   68.62740
61   60.95   70.86515
62   11.11   39.53669
71   11.11   39.53669
75   37.94   57.43867
76   75.71   66.38965
78   64.29   61.91416
83   78.07   70.86515
85    5.56   39.53669
89   11.11   39.53669
90   67.50   66.38965
91   71.49   66.38965
98   58.43   39.53669
```

Calculating Accuracy

1. R square represented as square of correlation between Actual values and predicted values.

```
> cor(actual_predict$actual,actual_predict$predict)
[1] 0.7674963
>
```

2. Min max Accuracy: Meanmaxaccuracy=min(actuals,predicted)/max(actual,predicted)

```
> mape= mean(abs((actual_predict$predicteds - actual_predict$actual))/ actual_predict$actual)*100
> mape
[1] 60.6191
> mape= mean(abs((actual_predict$predicteds - actual_predict$actual))/ actual_predict$actual)
> mape
[1] 0.606191
>
```

CONCLUSION: Thus we have implemented Multiple Linear Regressions successfully.

PRACTICAL 7

AIM: Practical of Logistics Regression .
Logistic regression predicts the probability of an event occurring. Models relationship between set of predictor variables Xi which are numeric anddichotomous categorical response variable Y.

In statistics, the logit function or the log-odds is the logarithm of the odds

p/(1 − p) where p is the probability.It is a type of function that creates a map of probability values from [0,1] to [− ∞, + ∞]

P(Y|X) is the probability of the event Y occurring, given event X.

Logit(P(Y|X)) = log(P(Y|X)/P(1-Y|X))

The logistic regression model is given by

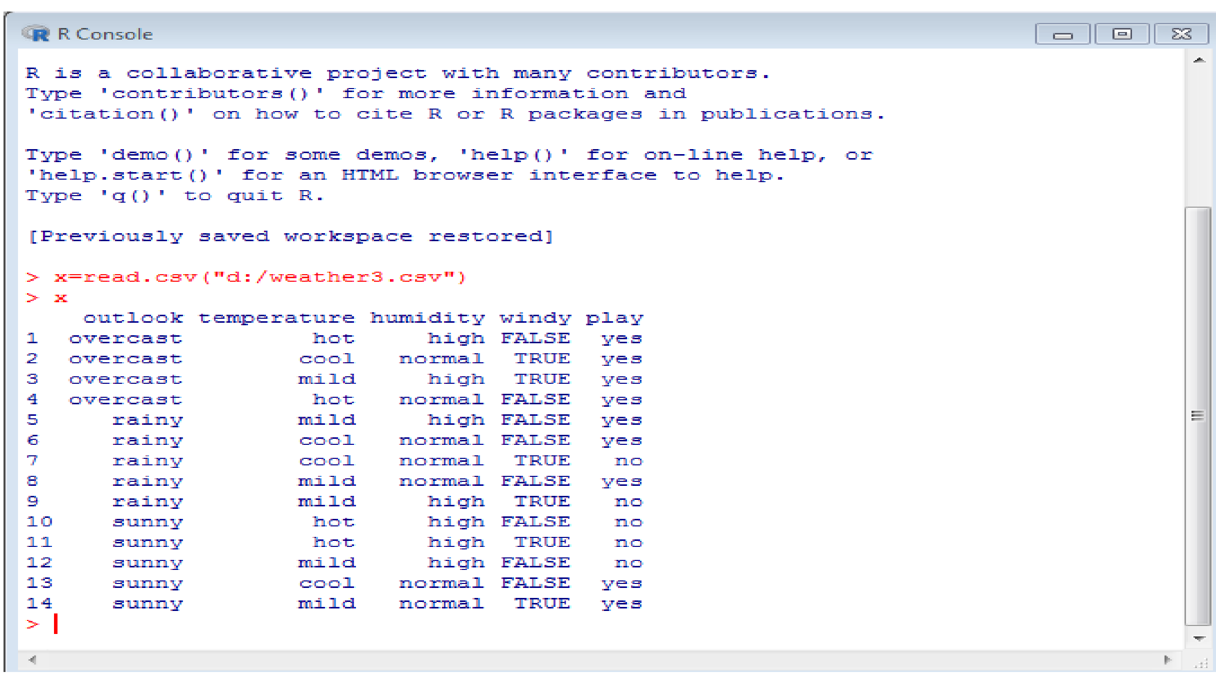
$P(Y|X)=(e^{\beta_0+\beta_1 X})/(1+e^{\beta_0+\beta_1 X})$

Where β_1Coefficient and $\beta_0$is the intercept.

IMPORT THE DATASET

> x=read.csv("d:/weather3.csv")

>x



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

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

[Previously saved workspace restored]

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

PREPROCESSING THE DATASET

Converting categorical string values to Dichotomous numeric variable

Converting humidity column.

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

> x

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

Converting the target variable Play to numeric values (dichotomous variables).

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

> x

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

Converting the variable windy to numeric values (dichotomous variables).

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

> x

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

PARTIONING DATASET

> s=sample(nrow(x),.7*nrow(x))

>x_tr=x[s,]

>x_test=x[-s,]

>nrow(x)

>nrow(x_tr)

>nrow(x_test)

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

DATA MODELING

Model 1

Testing the model with X as “windy” and Y as “play”

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

>lmod

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

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

Coefficients:
(Intercept)        windy
      20.57       -19.87

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

>summary(lmod)

```
> summary(lmod)

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

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

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 19

> |
```

The p value of windy calculated during the deployment of the logistic model is .998 which is far from .05 , so windy cannot be considered to a significant variable for classification of “weather dataset”

Model 2

Testing the model with X as "humidity" and Y as "play"

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

>summary(lmod)

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

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

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

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 18

> |
```

The p value of "temperature" calculated during the deployment of the logistic model is .998 which is much greater than .05, so windy cannot be considered to a significant variable for classification of "weather dataset"

Conclusion: This dataset cannot be accurately classified using logistic regression. Other classification models like decision tree, KNN can be used.

(2) <u>SECOND DATA SET:</u>

<u>Data:</u> Dataset consists of 104 records of students from a specific course appearing for 4 exams before the Final exam. The scores of all the 4 exams and the Final exam scores are collected.

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

>x2

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

Partitioning the dataset

> x=read.csv("d:/grade_logit.csv")

> s=sample(nrow(x),.7*nrow(x))

>x_tr=x[s,]

>x_test=x[-s,]

> x2_train=x[s,]

> x2_test=x[-s,]

Model 1

Testing the model with X as "Exam1" and Y as "Grade"

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

>summary(lmod2)

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

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

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

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

(Dispersion parameter for binomial family taken to be 1)

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

Number of Fisher Scoring iterations: 6
```

The p value of "Exam1" calculated during the deployment of the logistic model is .00211 which is less than .05, so Exam1 is a significant variable which can be used to predict the "Grade" of a student.

#Prediction output in in the form of probability of the student passing.

prediction=predict(lmod2,x2_test,type="response")

Prediction data 1’s and 0’s form

```
> prediction=predict(lmod2, x2_test,type="response")
> prediction
        1         2         4         8        11        16        17        22        27        30        31        35        36 
0.8871028 0.9479001 0.9823733 0.9768126 0.9479001 0.9322147 0.9651384 0.9479001 0.9543932 0.9003767 0.9695522 0.9823733 0.5944225 
       38        41        43        50        53        54        59        72        74        76        78        79        88 
0.8558962 0.8178351 0.5944225 0.9823733 0.9651384 0.5944225 0.9695522 0.9601112 0.8871028 0.9601112 0.9601112 0.9768126 0.5944225 
       90        91        92        93        97       102 
0.5944225 0.9695522 0.5944225 0.9405401 0.5944225 0.9543932 
> |
```

#Converting probability to 1s and 0s

>prediction=ifelse(p>.5,1,0)

>prediction

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

<u>PREDICTION MATRIX</u>

>table(x2_test$Grade,prediction)

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

> x2_test

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

#Printing actuals vs predicted values

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

>ac_pr

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

>vif(lmod2) // variable influence factor

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

CONCLUSION: Thus we have implemented Logistics Regression successfully

PRACTICAL NO 8

AIM: Practical of Hypothesis testing.

Hypothesis testing is used to infer the result of a hypothesis performed on sample data from a larger population. In hypothesis testing, an analyst tests a statistical sample, with the goal of accepting or rejecting a null hypothesis. The test tells the analyst whether or not his primary hypothesis is true.

A. One sample t test

The One Sample t Test determines whether the sample mean is statistically different from a known or hypothesized population mean. The One Sample t Test is a parametric test. This test is also known as: Single Sample t Test.

Data: We have 28 records of the time taken in minutes by employees of an Organization to complete a specific MIS report.

Null Hypothesis: There is no difference between the sample mean and the population mean which is taken as 100.

Alternate Hypothesis: There is a statistically significant difference exists between sample mean and population mean.

Step 1: First we createan Excel file and Enter the 28 values so that we can fine deviation from mean, Square of deviation, variance, T-value and standard deviation and save as **.**CSV file.

Data

C1
85.3
86.9
96.8
108.5
113.8
87.7
94.5
99.9
92.9
67.3
90.6
129.8
48.9
117.5
100.8

94.5
94.4
98.9
96
99.4
79.1
108.5
84.6
117.5
70
104.4
127.1
135

Excel File:

	A	B	C	D	E	
1		C1	Deviation	Deviation sqr		
2	1	85.3	-12.22142857	149.3633163		
3		86.9	-10.62142857	112.8147449		
4		96.8	-0.721428571	0.520459184		
5		108.5	10.97857143	120.5290306		
6		113.8	16.27857143	264.9918878		
7		87.7	-9.821428571	96.46045918		
8		94.5	-3.021428571	9.129030612		
9		99.9	2.378571429	5.657602041		
10		92.9	-4.621428571	21.35760204		
11		67.3	-30.22142857	913.3347449		
12		90.6	-6.921428571	47.90617347		
13		129.8	32.27857143	1041.906173		
14		48.9	-48.62142857	2364.043316		
15		117.5	19.97857143	399.1433163		
16		100.8	3.278571429	10.74903061		
17		94.5	-3.021428571	9.129030612		
18		94.4	-3.121428571	9.743316327		
19		98.9	1.378571429	1.900459184		
20		96	-1.521428571	2.314744898		
21		99.4	1.878571429	3.529030612		
22		79.1	-18.42142857	339.3490306		
23		108.5	10.97857143	120.5290306		
24		84.6	-12.92142857	166.9633163		
25		117.5	19.97857143	399.1433163		
26		70	-27.52142857	757.4290306		
27		104.4	6.878571429	47.3147449		
28		127.1	29.57857143	874.8918878		
29		135	37.47857143	1404.643316		
30		97.52143	calculate varianc	346.242398		
31					t value	-0.69214
32	populatio	100			system calculate stdev	18.94904
33	Diff in me	-2.47857				
34						

Mean: 97.52

Standard deviation: 18.94

Variance: 346.24

We will now verify the values calculated in Excel with the values calculated in R.

Step 2: Now we have to import Excel file (onetest.csv) type bellow command.

#datanew=read.csv("D:/onettest.csv")

#datanew

Output:

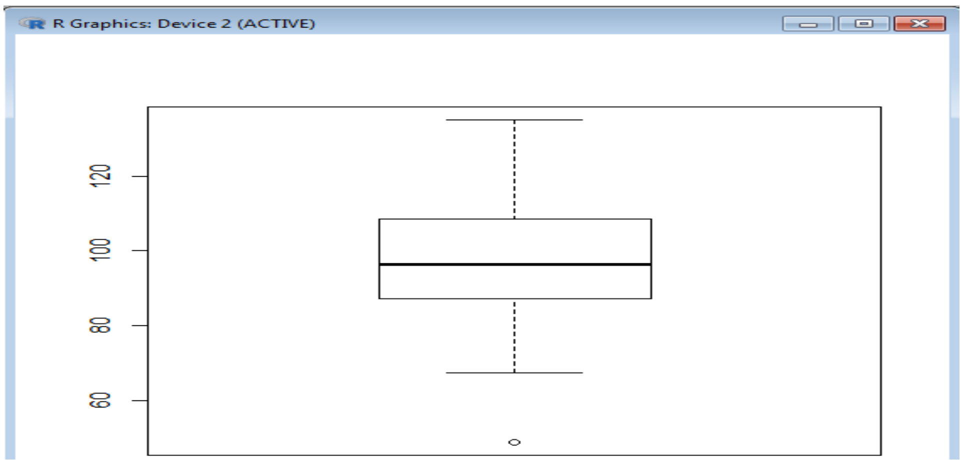
R Console

```
> datanew=read.csv("D:/onettest.csv")
> datanew
      C1
1   85.3
2   86.9
3   96.8
4  108.5
5  113.8
6   87.7
7   94.5
8   99.9
9   92.9
10  67.3
11  90.6
12 129.8
13  48.9
14 117.5
15 100.8
16  94.5
17  94.4
18  98.9
19  96.0
20  99.4
21  79.1
22 108.5
23  84.6
24 117.5
25  70.0
26 104.4
27 127.1
28 135.0
```

Step 3: After importing onetest.csv file we will plot Boxplot diagram type bellow command.

#boxplot(datanew)

Output:



Step 4: After that find mean of respective data.

m1=mean(datanew$C1)

#m1

Output:

```
> m1=mean(datanew$C1)
> m1
[1] 97.52143
```

Step 5: Now calculate the standard deviation.

#sd1=sd(datanew$C1)

#sd1

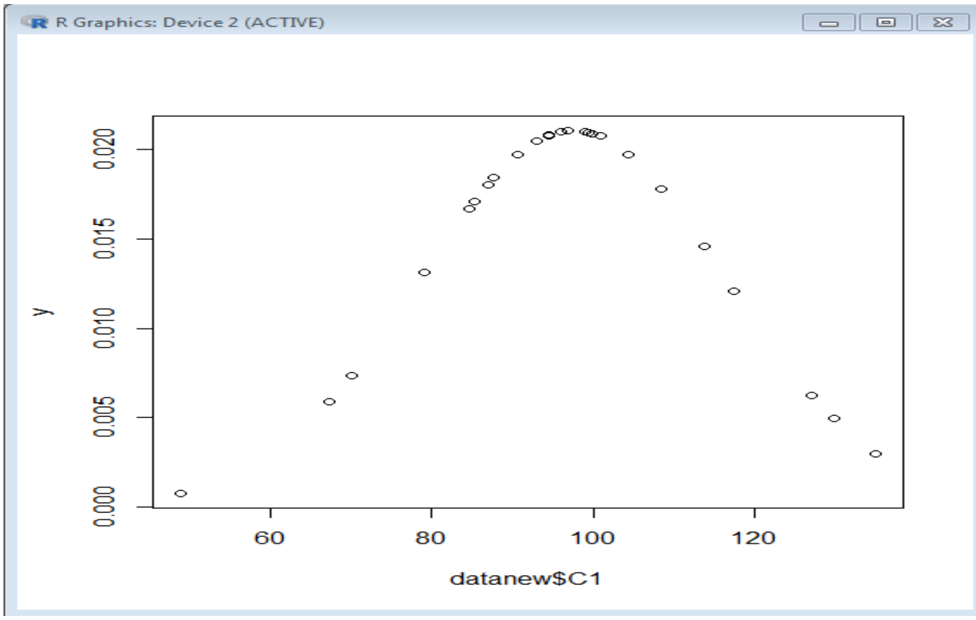
Output:

```
> sd1=sd(datanew$C1)
> sd1
[1] 18.94904
> mean1=mean(datanew$C1)
> mean1
[1] 97.52143
```

Step 6: Plot bell curve.

plot(datanew$C1)

Output:



The graph shows normal distribution of data favorable for doing T-test.

Step 7: At the end find T-Test value type following command.

#t.test(datanew$C1,alternative="greater",mu=100)

Output:

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

        One Sample t-test

data:  datanew$C1
t = -0.69214, df = 27, p-value = 0.7526
alternative hypothesis: true mean is greater than 100
95 percent confidence interval:
 91.4219      Inf
sample estimates:
mean of x
 97.52143
```

CONCLUSION: If p-value is less than .05 , we can accept alternate hypothesis, which says that there is a statistically significant difference between the sample mean and population mean.

In this case we accept the Null hypothesis which says that there is statistically no significant difference between the sample mean and population mean.

B. Hypothesis testing using two sampled t-test.

The **unpaired two-samples t-test** is used to compare the **mean** of two independent groups.

Step 1: Create excel file for two sample t-test.

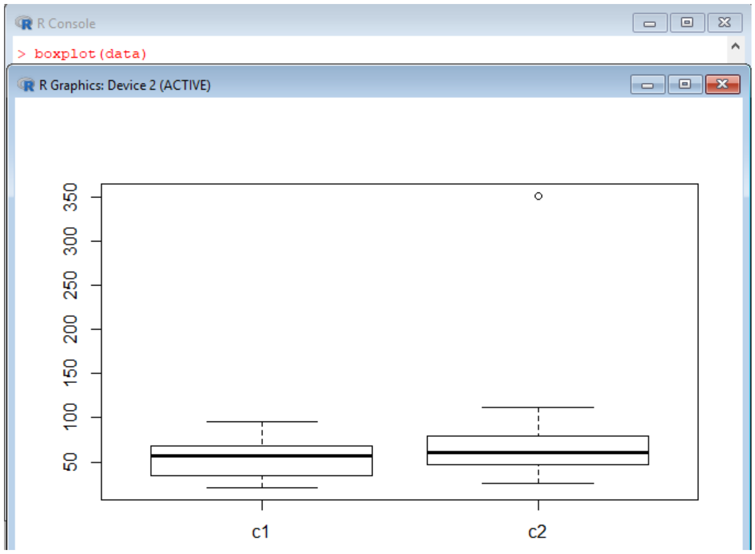
	A	B	C	D	E	F	G
1	c1	c2					
2	75.6	52					
3	56.5	75					
4	21	67					
5	34.2	112					
6	68.5	351					
7	96.1	67					
8	65.2	84					
9	52	39					
10	67	49					
11	82	26		tvalue	1.300132		
12	32	74					
13	21	61					
14	59	83					
15	69	46					
16	51	57					
17	34	36					
18	46	27					
19	75	94					
20	32	53					
21				stdevC1	21.38726	24.07446	283.4626
22	mean1	54.58421		stdevC2	70.20238	259.3881	
23	mean2	76.47368	21.88947			den	16.83635

Step 2: Generate two sample files on R console.

R Console

```
> data=read.csv("F:/tycs/test.csv")
> data
     c1  c2
1  75.6  52
2  56.5  75
3  21.0  67
4  34.2 112
5  68.5 351
6  96.1  67
7  65.2  84
8  52.0  39
9  67.0  49
10 82.0  26
11 32.0  74
12 21.0  61
13 59.0  83
14 69.0  46
15 51.0  57
16 34.0  36
17 46.0  27
18 75.0  94
19 32.0  53
```

Step 3: After importing test.csv file we will plot Boxplot diagram type bellow command.



Step 4: After that find mean of respective data.

```
> m1=mean(data$c1)
> m1
[1] 54.58421
> m2=mean(data$c2)
> m2
[1] 76.47368
>
```

Step 5: Now calculate the standard deviation.

```
> s1=sd(data$c1)
> s1
[1] 21.38726
> s2=sd(data$c2)
> s2
[1] 70.20238
```

Step 7: At the end find T-Test value type following command.

```
> t.test(data$c1,data$c2,alternative="greater")

        Welch Two Sample t-test

data:  data$c1 and data$c2
t = -1.3001, df = 21.313, p-value = 0.8963
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -50.84091       Inf
sample estimates:
mean of x mean of y
 54.58421  76.47368
```

CONCLUSION: We performed a one-tailed t-test to check whether there is a difference between the means of two samples and whether sample s1 mean is greater than sample s2 mean. But the p-value is more than .05 indicating that the null hypothesis holds.

PRACTICAL NO 9

AIM: Analysis of Variance
Analysis of Variance (ANOVA) is a parametric statistical technique used to compare datasets. This technique was invented by R.A. Fisher, and is thus often referred to as Fisher's ANOVA, as well. It is similar in application to techniques such as t-test and z-test, in that it is used to compare means and the relative variance between them. However, analysis of variance (ANOVA) is best applied where more than 2 populations or samples are meant to be compared.

Data: 3 groups of data are taken considerably different from each other.

 # CREATE THE DATA IN TO THREE GROUPS

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

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

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

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

>cg

```
> group1=c(2,3,7,2,6)
> group2=c(10,8,7,5,10)
> group3=c(10,13,14,13,15)
> cg=data.frame(cbind(group1,group2,group3))
> cg
  group1 group2 group3
1      2     10     10
2      3      8     13
3      7      7     14
4      2      5     13
5      6     10     15
```

#TO PRINT THE SAME DATA INTO .CSV FORMAT

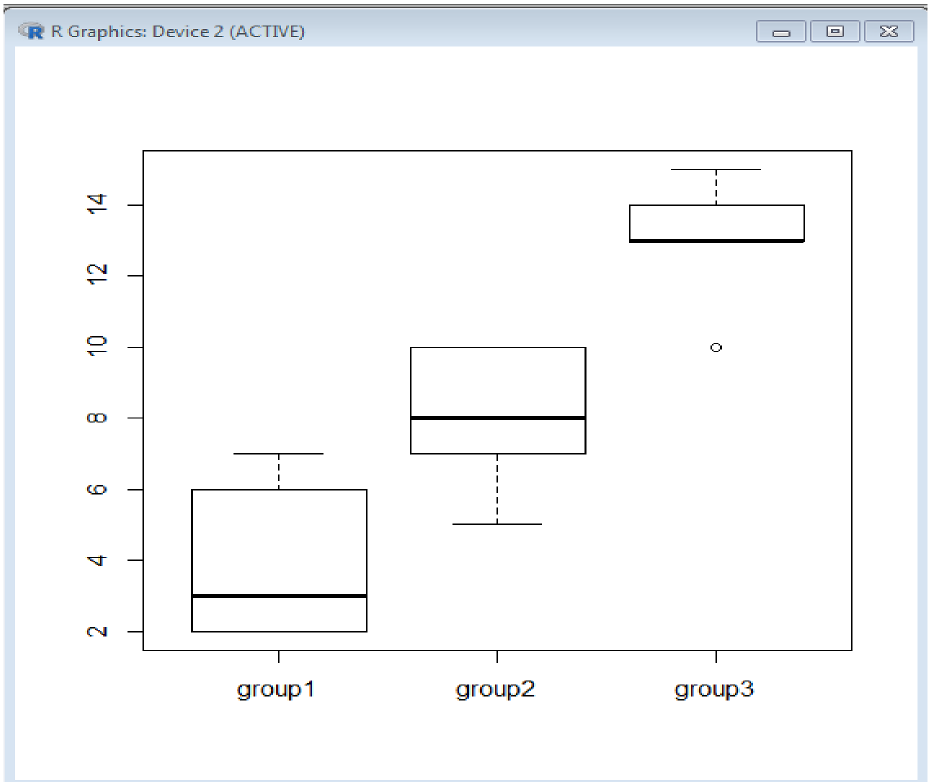
>write.csv(cg,"D:/cg.csv")

>summary(cg)

```
> write.csv(cg,"D:/cg.csv")
> summary(cg)
     group1       group2       group3
 Min.   :2   Min.   : 5   Min.   :10
 1st Qu.:2   1st Qu.: 7   1st Qu.:13
 Median :3   Median : 8   Median :13
 Mean   :4   Mean   : 8   Mean   :13
 3rd Qu.:6   3rd Qu.:10   3rd Qu.:14
 Max.   :7   Max.   :10   Max.   :15
>
```

#TO PRINT THE BOXPLOT

>boxplot(cg)



#TO PRINT THE DATA INTO STACK FORMAT

>stacked_g=stack(cg)

>stacked_g

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

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

>summary(av)

```
> av=aov(values~ind,data=stacked_g)
> summary(av)
            Df Sum Sq Mean Sq F value   Pr(>F)    
ind          2  203.3   101.7   22.59 8.54e-05 ***
Residuals   12   54.0     4.5                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> 
```

Df(Degree of freedom)= number of groups-1 [3-1=2]

Residuals= number of samples- number of groups[15-3=12]

Null hypothesis: The groups have no difference from each other.

Alternate hypothesis: There is considerable difference in variation between the groups.

The p value is <.05 which indicates that we can accept the alternate hypothesis which says that there is a statistically significant difference between the groups.

2. TAKE ANOTHER DATASET AND WORK ON THAT.

CREATE THE DATA IN TO THREE GROUPS

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

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

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

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

> cg1

```
> g1=c(29,30,31,31,29)
> g2=c(28,29,27,30,29)
>  g3=c(25,28,29,27,29)
> cg1=data.frame(cbind(g1,g2,g3))
> cg1
  g1 g2 g3
1 29 28 25
2 30 29 28
3 31 27 29
4 31 30 27
5 29 29 29
```

#TO PRINT THE DATA INTO STACK FORMAT

>stacked_g=stack(cg1)

>stacked_g

```
> stacked_g=stack(cg1)
> stacked_g
   values ind
1      29  g1
2      30  g1
3      31  g1
4      31  g1
5      29  g1
6      28  g2
7      29  g2
8      27  g2
9      30  g2
10     29  g2
11     25  g3
12     28  g3
13     29  g3
14     27  g3
15     29  g3
```

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

>summary(av1)

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

Conclusion: In this case also we accept the alternate hypothesis as it is proved that there exists considerable statistical difference between the groups.

Assessment No. 10

Aim: Practical of Decision tree

Steps:

❖ **Install 'rpart' and 'tree' packages in rstudio.**

```
> install.packages("rpart")
Installing package into 'C:/Users/admin/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/rpart_4.1-13.zip'
Content type 'application/zip' length 950583 bytes (928 KB)
downloaded 928 KB

package 'rpart' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\admin\AppData\Local\Temp\RtmpITaQNH\downloaded_packages
> install.packages("tree")
Installing package into 'C:/Users/admin/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/tree_1.0-39.zip'
Content type 'application/zip' length 177754 bytes (173 KB)
downloaded 173 KB

package 'tree' successfully unpacked and MD5 sums checked
```

Data: Dataset consists of sample test data containing 4 features which enable the system to take decision whether a golf player can play that day or not. The decision is contained in a column or the target class or output variable "play golf"

❖ **Create an excel data save it with .csv extension.**

outlook	temp	humidity	windy	play golf
rainy	hot	high	FALSE	no
rainy	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
sunny	mild	high	FALSE	yes
sunny	cool	normal	FALSE	yes
sunny	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
rainy	mild	high	FALSE	no
rainy	cool	normal	FALSE	yes
sunny	mild	normal	FALSE	yes
rainy	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
sunny	mild	high	TRUE	no

❖ **Read excel data in rstudio**

```
> x=read.csv("D:/TYCS46/weather1.csv")
>x

   Outlook Temp Humidity windy playgolf
1     Rainy  Hot     high FALSE       no
2     Rainy  Hot     high  TRUE       no
3  Overcast  Hot     high FALSE      yes
4     Sunny mild     high FALSE      yes
5     Sunny Cool   normal FALSE      yes
6     Sunny Cool   normal  TRUE       no
7  Overcast Cool   normal  TRUE      yes
8     Rainy mild     high FALSE       no
9     Rainy Cool   normal FALSE      yes
10    Summy mild   normal FALSE      yes
11    Rainy mild   normal  TRUE      yes
12 Overcast mild     high  TRUE      yes
13 Overcast  hot   normal FALSE      yes
14    Sunny mild     high  TRUE       no
```

- **Create sample partition of the excel data**
  >sample_weather=sample(nrow(x),.7*nrow(x))

- **Create a weather partition for training**

  weather_tr=x[sample_weather,]

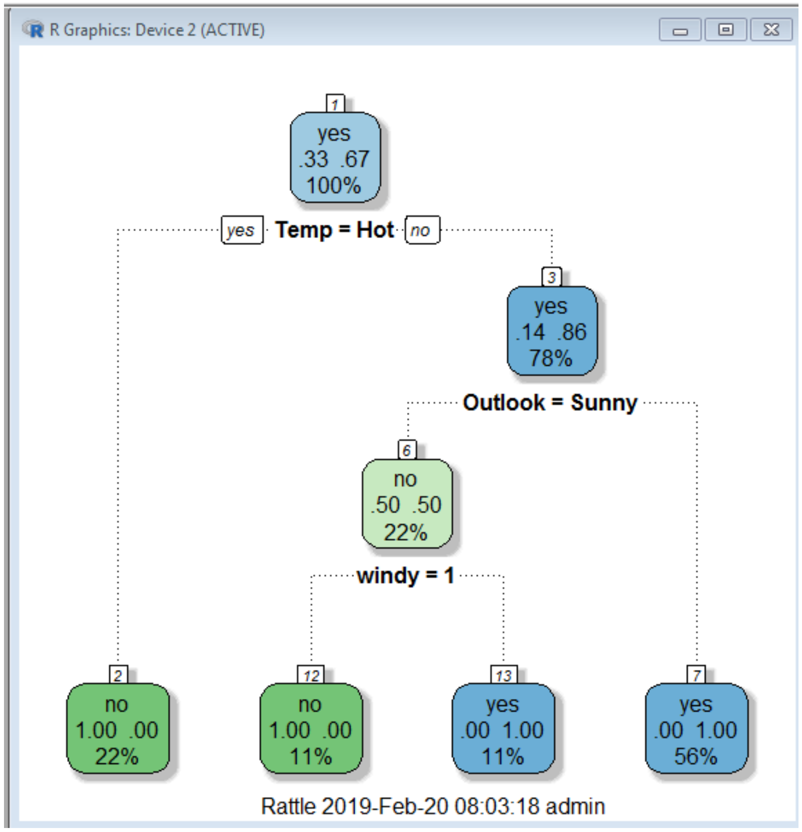
- **Create a weather partition for testing**
  >weather_test=x[-sample_weather,]
  >weather_test

- **Call rpart package**

  >library("rpart")
  >library(rattle) (if not installed install from toolbox use USA CA 1)

- **Plot tree**
  >dtreemod=rpart(playgolf~.,data=weather_tr,method="class",control=rpart.control(minsplit=1,minbucket=1))
  >fancyRpartPlot(dtreemod)

Output:



❖ **Predict Tree:**

```
>p=predict(dtreemod,weather_test,type="class")

>weather_test
   Outlook Temp Humidity windy playgolf
3  Overcast  Hot     high FALSE      yes
4     Sunny mild     high FALSE      yes
8     Rainy mild     high FALSE       no
13 Overcast  hot   normal FALSE      yes
14    Sunny mild     high  TRUE       no
```

#Accuracy calculation

```
>table(weather_test$playgolf,p)
p
no yes
no   1   1
yes  1   2
```

Conclusion: The decision tree is created using the Weather dataset and then used to predict the class of the test dataset with 60% accuracy

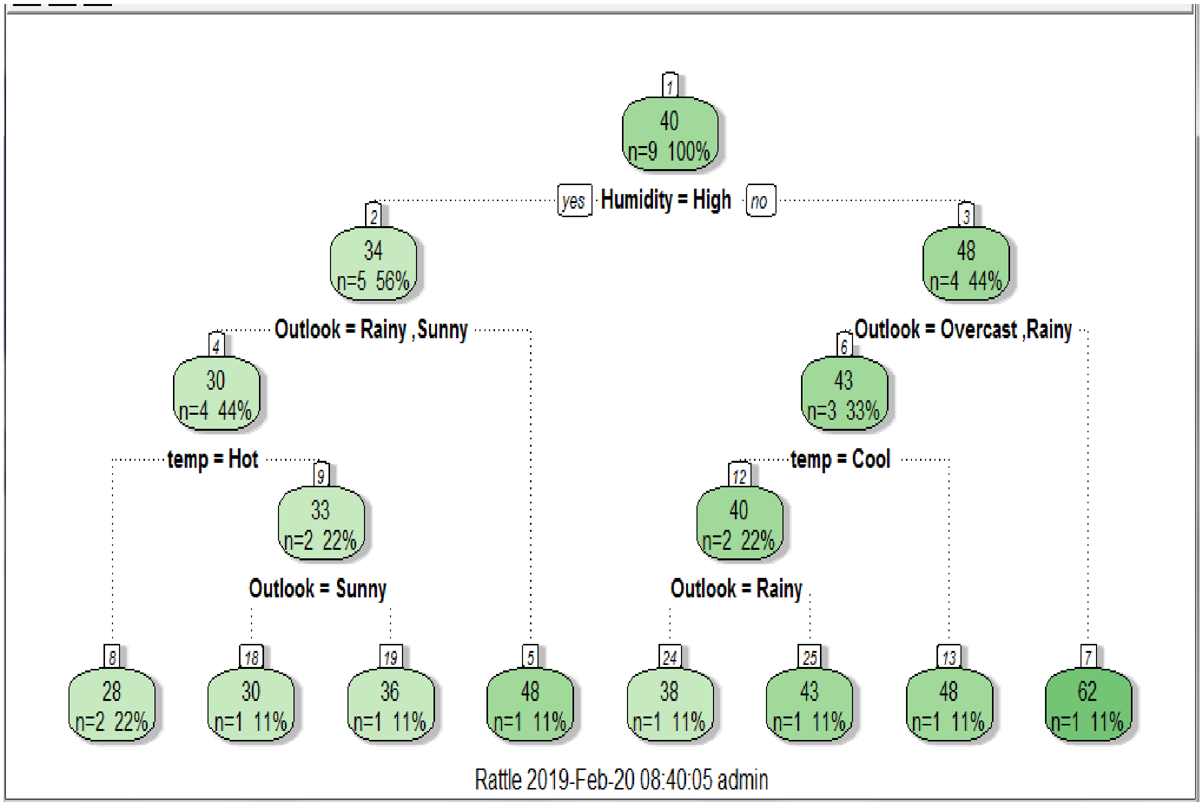
Regression Tree:

The regression trees are created in case where the response variable is either continuous or numeric, but not categorical.Regression trees can be applied in case of prices, quantities, or data involving quantities etc.

Data: Consists of the weather dataset with the same features but the feature “play Golf” which is categorical is replaced by Hours_played which is numeric variable.

```
>x=read.csv("D:/TYCS46/weather3.csv")
>x
Outlook  temp Humidity  Windy Hours_played
1    Rainy   Hot    High  FALSE          26
2    Rainy   Hot    High   TRUE          30
3  Overcast   Hot    High  FALSE          48
4    Sunny  Mild    High  FALSE          46
5    Sunny  Cool   Normal  FALSE          62
6  Overcast  Cool   Normal   TRUE          43
7    Rainy  Mild    High  FALSE          36
8    Rainy  Cool   Normal  FALSE          38
9    Sunny  Mild   Normal  FALSE          48
10   Rainy  Mild  Normal    TRUE          48
11 Overcast  Mild    High   TRUE          62
12 Overcast   Hot   Normal  FALSE          44
13   Sunny  Mild    High   TRUE          30

> s=sample(nrow(x),.7*nrow(x))
>weather_tr=x[s,]
>weather_test=x[-s,]
>dtreemod=rpart(Hours_played~.,data=weather_tr,method="anova",control=)

>dtreemod=rpart(Hours_played~.,data=weather_tr,method="anova",control=rpart.control(minspli
t=1,minbucket=1))
>fancyRpartPlot(dtreemod)
```

Output:



Prediction:

```
>actuals_preds<- data.frame(cbind(actuals=weather_test$Hours_played,predicts=p))
Warning message:
In cbind(actuals = weather_test$Hours_played, predicts = p) :
number of rows of result is not a multiple of vector length (arg 1)
>actuals_preds
actuals predicts
3     46     1
4     48     2
8     62     2
13    44     2
14    46     1
```

Conclusion: The Regression tree is created using the Weather dataset.