Roll No: - 19019

R.D. & S. H. National College & S. W.A. Science College

Bandra, Mumbai – 400050.

Department of Computer Science

CERTIFICATE

This is to certify that Mr./Ms. AVINASH RAJKUMAR KAURAN of TY BSc CS class (VI Semester) has satisfactorily completed 9 Practicals, in the subject of Data Science as a part of B.Sc. Degree Course in Computer Science during the academic year 20 21 – 20 22 .

Date of Certification:

Faculty Incharge

Head,
Department Computer Science

Signature of Examiner

Roll No: - 19019

INDEX

SR. NO	DATE	AIM	PAGE NO	
1	04/01/2022	Practical of Data collection, Data curation and management for Unstructured data (NoSQL)	3	
2	11/01/2022	Practical of Data collection, Data curation and management for Large-scale Data system (such as MongoDB)	7	
3	18/01/2022	Practical of Principal Component Analysis.	15	
4	25/01/2022	Practical of Clustering.	22	
5	01/02/2022	Practical of Time-series forecasting	28	
6	08/02/2022	Practical of Simple/Multiple Linear Regression	32	
7	15/02/2022	Practical of Logistics Regression	41	
8	22/02/2022	Practical of Hypothesis testing	45	
9	08/03/2022	Practical of Decision Tree	46	

Practical No: 1

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

- **Package Installation**
  Code:install.package('sofa')

- **Create connection object**
  Syntax:con_obj<-Cushion$new()
  Code: x<-Cushion$new()

- **Check whether object created**

Syntax:con_obj$ping()
Code:x$ping()
Output:

```
> x$ping()
$couchdb
[1] "Welcome"

$version
[1] "2.3.0"

$git_sha
[1] "07ea0c7"

$uuid
[1] "a735a732103d52cecf0fec44e6fad187"

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

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


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

- **Create database ty_prac**

Syntax:Code:db_create(con_obj, dbname = db_name)
Code:db_create(x,dbname = 'ty_prac')

```
> db_create(x,dbname = 'ty_prac')
$ok
[1] TRUE
```

- **Show all databases**

Syntax:db_list(con_obj)
Code:db_list(x)
Output:

```
> db_list(x)
[1] "papibhai" "ty"       "ty_prac"
```

- **Create json doc**

Syntax:doc_create(con_obj,var_name,dbname = db_name,docid =doc_id)

Code:doc_create(x,doc3,dbname = "ty_prac",docid = "a_3")

```
> doc3<-'{"rollno":"03","name":"xyz","GRADE":"B","REMARK":"PASS"}'
> doc_create(x,doc3,dbname = "ty_prac",docid = "a_3")
$ok
[1] TRUE

$id
[1] "a_3"

$rev
[1] "1-f13d2a583fc7fd0d645d421014a295b2"
```

- **Changes Feed**

Syntax:db_changes(con_obj, db_name)
Code:db_changes(x,"ty_prac")
Output:

```
> db_changes(x,"ty_prac")
$results
$results[[1]]
$results[[1]]$seq
[1] "1-g1AAAAF1eJzLYWBg4MhgTmEQTM4vTc5ISXLIyU9OzMnILy7JAUoxJTIkyf___z8rkQGPoiQFIJlkD1KXw
ZzImAvksRuYWRgnm6Vg04PPJAeQSfGEbUwAqasnqC6PBUgyNAApoNL5xKhdAFG7nxi1ByBq7xOj9gFELci9WQDye
Hky"

$results[[1]]$id
[1] "a_3"

$results[[1]]$changes
$results[[1]]$changes[[1]]
$results[[1]]$changes[[1]]$rev
[1] "1-f13d2a583fc7fd0d645d421014a295b2"
```

- **Search for id > null so all docs will display**

Syntax:db_query(con_obj,dbname = db_name, selector = list(attribute=val))$docs
Code:db_query(x,dbname = "ty_prac", selector = list('_id'=list('$gt'=NULL)))$docs

```
> db_query(x,dbname = "ty_prac", selector = list('_id'=list('$gt'=NULL)))$docs
[[1]]
[[1]]$`_id`
[1] "a_1"

[[1]]$`_rev`
[1] "1-be7c98bddf8ea7c46f4f401ff387593d"

[[1]]$rollno
[1] "01"

[[1]]$name
[1] "ABC"

[[1]]$GRADE
[1] "A"


[[2]]
[[2]]$`_id`
[1] "a_2"

[[2]]$`_rev`
[1] "1-1ddcb45704c37893389b050ddbdc440a"

[[2]]$rollno
[1] "02"

[[2]]$name
[1] "PQR"

[[2]]$GRADE
[1] "A"


[[3]]
[[3]]$`_id`
```

- **Search for students with grade is A**

Syntax:db_query(con_obj,dbname = db_name,selector = list(attribute=val))$docs
Code:db_query(x,dbname = "ty",selector = list(GRADE="A"))$docs
Output:

```
> db_query(x,dbname = "ty_prac",selector = list(GRADE="A"))$docs
[[1]]
[[1]]$`_id`
[1] "a_1"

[[1]]$`_rev`
[1] "1-be7c98bddf8ea7c46f4f401ff387593d"

[[1]]$rollno
[1] "01"

[[1]]$name
[1] "ABC"

[[1]]$GRADE
[1] "A"


[[2]]
[[2]]$`_id`
[1] "a_2"

[[2]]$`_rev`
[1] "1-1ddcb45704c37893389b050ddbdc440a"

[[2]]$rollno
[1] "02"

[[2]]$name
[1] "PQR"

[[2]]$GRADE
[1] "A"
```

- **Search for students with remark =pass**

Syntax:db_query(con_obj,dbname = db_name, selector = list(attribute=val))$docs
Code:db_query(x,dbname = "ty",selector = list(REMARK="PASS"))$docs

```
> db_query(x,dbname = "ty_prac",selector = list(REMARK="PASS"))$docs
[[1]]
[[1]]$`_id`
[1] "a_3"

[[1]]$`_rev`
[1] "1-f13d2a583fc7fd0d645d421014a295b2"

[[1]]$rollno
[1] "03"

[[1]]$name
[1] "xyz"

[[1]]$GRADE
[1] "B"

[[1]]$REMARK
[1] "PASS"
```

- **Return only certain fields where rollno>2**

Syntax:db_query(con_obj,dbname = db_name,selector = list(attribute=val),fields=c(attributes))$docs
Code:db_query(x,dbname = "ty",selector = list(rollno=list('$gt'='02')),fields=c("name","GRADE"))$docs
Output:

```
> db_query(x,dbname = "ty_prac",selector = list(rollno=list('$gt'='02')),fields=c("name"
,"GRADE"))$docs
[[1]]
[[1]]$name
[1] "xyz"

[[1]]$GRADE
[1] "B"
```

- **Convert the result of a query into a data frame using jsonlite**

Code:library("jsonlite")
res<-db_query(x,dbname = "ty",selector =
list('_id'=list('$gt'=NULL)),fields=c("name","rollno","GRADE","REMARK"),as="json")

- **Display json doc**

Syntax:fromJSON(var_name)$docs
Code:fromJSON(res)$docs
Output:

```
> res<-db_query(x,dbname = "ty_prac",selector = list('_id'=list('$gt'=NULL)),fields=c("n
ame","rollno","GRADE","REMARK"),as="json")
> #display json doc
> fromJSON(res)$docs
  name rollno GRADE REMARK
1  ABC     01     A   <NA>
2  PQR     02     A   <NA>
3  xyz     03     B   PASS
```

- **Deleting entry**

Syntax:doc_delete(cushion,dbname,docid)
Code:doc_delete(x,dbname = "ty",docid = "a_2")
Output:

```
> doc_delete(x,dbname = "ty_prac",docid = "a_2")
$ok
[1] TRUE

$id
[1] "a_2"

$rev
[1] "2-82f1879cc7d73bef5574cc5cdf7c4094"

> doc_get(x,dbname = "ty_prac",docid = "a_2")
Error: (404) - deleted
```

- **Updating Entry**

Syntax:doc_update(con_obj, dbname = db_name, doc=var_name,docid=doc_id,rev = value)
Code:doc_update(x,dbname = "ty",doc=doc2,docid="a_3",rev = "3-
b1fb56db955b142c6efd3b3c52fe9e1b")
Output:

```
> doc2<-'{"name":"Sdrink","beer":"TEST","note":"yummy","note2":"yay"}'
> doc_update(x,dbname = "ty_prac",doc=doc2,docid="a_3",rev = "1-f13d2a583fc7fd0d645d4210
14a295b2")
$ok
[1] TRUE

$id
[1] "a_3"
```

Practical No: 2

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

Create Database in MongoDB

Syntax: use database_name

Code: use practical

Output:

```
> use practical
switched to db practical
```

Drop Database in MongoDB

Code: db.dropDatabase()

Output:

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

Create Collection in MongoDB

Method 1: Creating the Collection in MongoDB on the fly

Syntax: db.collection_name.insert({key:value, key:value…})

Code: db.collection1.insert({id:001, Name:"Rajat"})

Output:

```
> db.collection1.insert({id:001, Name:"Rajat"})
WriteResult({ "nInserted" : 1 })
```

To check whether the collection is created successfully, use the following command.

Code: show collections

Output:

```
> show collections
collection1
```

Method 2: Creating collection with options before inserting the documents

Syntax:db.createCollection(name, options)

Code:db.createCollection("collection2")

Output:

```
> db.createCollection("collection2")
{ "ok" : 1 }
```

Drop collection in MongoDB

Syntax:db.collection_name.drop()

Code:db.collection2.drop()

Output:

```
> db.collection2.drop()
true
```

MongoDB Insert Document

Syntax: db.collection_name.insert()

Code: db.collection1.insert({ id: 002, name:"Raj", course:[{name:"CS", duration:7}, {name:"Java", duration:5}]})

Output:

```
> db.collection1.insert({ id: 002, name:"Raj", course:[{name:"CS", duration:7}, {name:"Java", duration:5}]})
WriteResult({ "nInserted" : 1 })
```

MongoDB Example: Insert Multiple Documents in collection

Code:varins = [

{"StudentID" : 100, "Name" : "Rajaa"} ,

{"StudentID" : 101,"Name" : "Raju"}];

db.collections1.insert(ins);

Output:

```
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ]
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

MongoDB Query Document using find() method

Syntax: db.collection_name.find()

Code:db.collection1.find()

Output:

```
> db.collection1.find()
{ "_id" : ObjectId("5c4980d0e65a457e05c82e88"), "id" : 1, "Name" : "Rajat" }
```

Querying all the documents in JSON format

Code :db.collection1.find().forEach(printjson); or db.collection1.find().pretty();

Output:

```
{ "_id" : ObjectId("5c4980d0e65a457e05c82e88"), "id" : 1, "Name" : "Rajat" }
{
        "_id" : ObjectId("5c498547e65a457e05c82e89"),
        "id" : 2,
        "name" : "Raj",
        "course" : [
                {
                        "name" : "CS",
                        "duration" : 7
                },
                {
                        "name" : "Java",
                        "duration" : 5
                }
        ]
}
```

Query Document based on the criteria

Equality Criteria:

Code:db.collection1.find({Name : "Rajat"}).pretty()

Output:

```
{ "_id" : ObjectId("5c4980d0e65a457e05c82e88"), "id" : 1, "Name" : "Rajat" }
```

Greater Than Criteria:

Syntax: db.collection_name.find({"field_name":{$gt:criteria_value}}).pretty()

Code:db.collection1.find({"id":{$gt:1}}).pretty()

Output:

```
{
        "_id" : ObjectId("5c498547e65a457e05c82e89"),
        "id" : 2,
        "name" : "Raj",
        "course" : [
                {
                        "name" : "CS",
                        "duration" : 7
                },
                {
                        "name" : "Java",
                        "duration" : 5
                }
        ]
}
```

Greater Than Criteria:

Syntax: db.collection_name.find({"field_name":{$gt:criteria_value}}).pretty()

Code: db.collection1.find({"id":{$lt:2}}).pretty()

Output:

```
{ "_id" : ObjectId("5c4980d0e65a457e05c82e88"), "id" : 1, "Name" : "Rajat" }
```

Not Equals Criteria:

Syntax: db.collection_name.find({"field_name":{$ne:criteria_value}}).pretty()

Code:db.collection1.find({"id":{$ne:2}}).pretty()

Output:

```
{ "_id" : ObjectId("5c4980d0e65a457e05c82e88"), "id" : 1, "Name" : "Rajat" }
```

Greater than equals Criteria:

Syntax: db.collection_name.find({"field_name":{$gte:criteria_value}}).pretty()

Code:db.collection1.find({"id":{$gte:2}}).pretty()

Output:

```
{
        "_id" : ObjectId("5c498547e65a457e05c82e89"),
        "id" : 2,
        "name" : "Raj",
        "course" : [
                {
                        "name" : "CS",
                        "duration" : 7
                },
                {
                        "name" : "Java",
                        "duration" : 5
                }
        ]
}
```

Less than equals Criteria:

Syntax: db.collection_name.find({"field_name":{$lte:criteria_value}}).pretty()

Code:db.collection1.find({"id":{$lte:2}}).pretty()

Output:

```
{ "_id" : ObjectId("5c4980d0e65a457e05c82e88"), "id" : 1, "Name" : "Rajat" }
{
        "_id" : ObjectId("5c498547e65a457e05c82e89"),
        "id" : 2,
        "name" : "Raj",
        "course" : [
                {
                        "name" : "CS",
                        "duration" : 7
                },
                {
                        "name" : "Java",
                        "duration" : 5
                }
        ]
}
```

MongoDB – Update Document in a Collection.

Syntax: db.collection_name.update(criteria,update_data)

Code:db.collection1.update({"Name":"Rajat"},{$set:{"Name":"Jat"}})

Output:

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Updating Document using save() method

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

Code:db.collection1.save({_id:ObjectId("5c4980d0e65a457e05c82e88"),"Name":"Jat"})

Output:

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

MongoDB Delete Document from a Collection

Syntax:db.collection_name.remove(delete_criteria)

Code: db.collection1.remove({"id":2});

Output:

```
WriteResult({ "nRemoved" : 1 })
```

MongoDB Delete Only OneDocument from a Collection

Syntax:db.collection_name.remove(delete_criteria,justOne)

Code:db.collection1.remove({"Name":"Jat"},1)

Output:

```
WriteResult({ "nRemoved" : 1 })
```

Remove all Documents

Code: db.collection_name.remove({})

Output:

```
WriteResult({ "nRemoved" : 3 })
```

MongoDB Projection

Syntax:db.collection_name.find({},{field_key:1or0})

Code:db.collection1.find({},{"name":1})

Output:

```
{ "_id" : ObjectId("5c4ec79631bfe13e1dcf17b4"), "name" : "Rajat" }
{ "_id" : ObjectId("5c4ec79731bfe13e1dcf17b5"), "name" : "Raj" }
{ "_id" : ObjectId("5c4ec79731bfe13e1dcf17b6"), "name" : "Raju" }
```

MongoDB – limit( ) and skip( ) method

The limit() method in MongoDB

Syntax:db.collection_name.find().limit(number_of_documents)

Code:db.collection1.find({id:{$gt:2}}).limit(1).pretty()

Output:

```
{
        "_id" : ObjectId("5c4ec79731bfe13e1dcf17b6"),
        "id" : 3,
        "name" : "Raju",
        "age" : 43
}
```

MongoDB Skip() Method

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

Code:db.collection1.find({id:{$gt:0}}).limit(1).skip(1).pretty()

Output:

```
{
        "_id" : ObjectId("5c4ec79731bfe13e1dcf17b5"),
        "id" : 2,
        "name" : "Raj",
        "age" : 42
}
```

MongoDB sort() method

Syntax:db.collecttion_name.find().sort({field_key:1or-1})

Code:db.collection1.find().sort({id:-1}) (1 for Ascending and -1 for Descending)

Output:

```
{ "_id" : ObjectId("5c4ec79731bfe13e1dcf17b6"), "id" : 3, "name" : "Raju", "age" : 43 }
{ "_id" : ObjectId("5c4ec79731bfe13e1dcf17b5"), "id" : 2, "name" : "Raj", "age" : 42 }
{ "_id" : ObjectId("5c4ec79631bfe13e1dcf17b4"), "id" : 1, "name" : "Rajat", "age" : 42 }
```

MongoDB Indexing Tutorial with Example

How to create index in MongoDB

Syntax:db.collection_name.createIndex({field_name:1or-1})(1 for Ascending and -1 for Descending)

Code:db.collection1.createIndex({id:-1})

Output:

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

MongoDB – Finding the indexes in a collection

Syntax:db.collection_name.getIndexes()

Code:db.collection1.getIndexes()

Output:

```
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "practical.collection1"
        },
        {
                "v" : 2,
                "key" : {
                        "id" : -1
                },
                "name" : "id_-1",
                "ns" : "practical.collection1"
        }
]
```

MongoDB – Drop indexes in a collection

Dropping a specific index:

Syntax:db.collection_name.dropIndex({index_name:1})

Code:db.collection1.dropIndex({name:-1})

Output:

```
{ "nIndexesWas" : 3, "ok" : 1 }
```

Dropping all the indexes:

Code:db.collection1.dropIndexes()

Output:

```
{
        "nIndexesWas" : 2,
        "msg" : "non-_id indexes dropped for collection",
        "ok" : 1
}
```

Practical No: 3

Aim: - Practical of Principal Component Analysis.

Code:

data("iris")
head(iris)

Output:

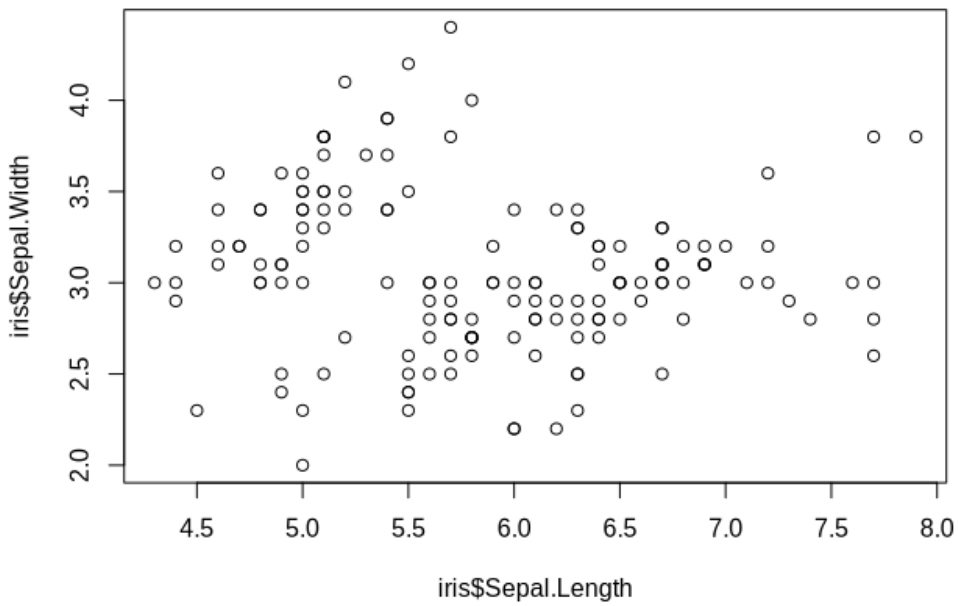
```
> data("iris")
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          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
>
```

Code:
summary(iris)

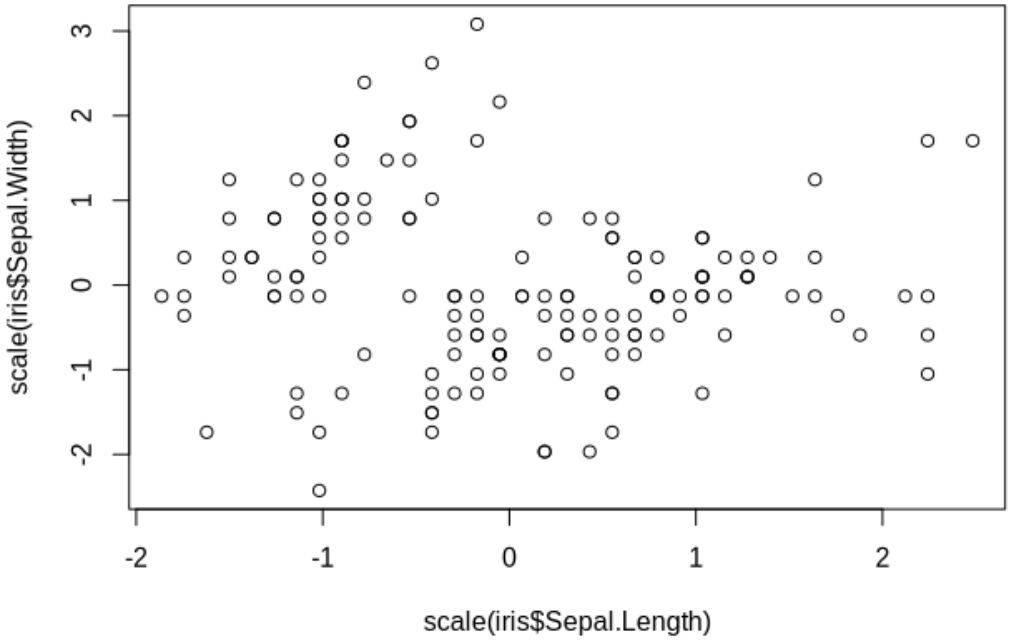
Output:

Code:
myPR <- prcomp(iris[, -5], scale = T)
plot(iris$Sepal.Length, iris$Sepal.Width)

Output:



Code:
plot(scale(iris$Sepal.Length), scale(iris$Sepal.Width))



Code:
myPR

Output:

```
> myPR
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
> 
```

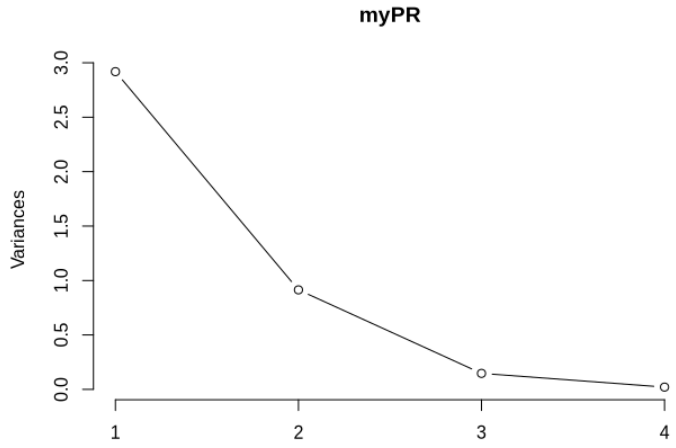
Code:
summary(myPR)

Output:

```
> summary(myPR)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
> 
```

Code:
plot(myPR, type='l')

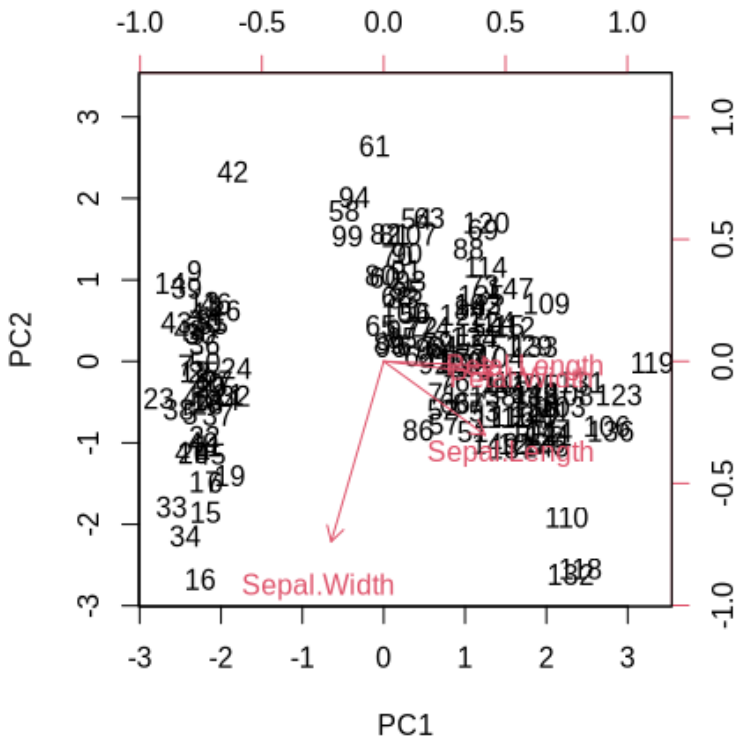
Output:



Code:

```
biplot(myPR, scale = 0)
```

Output:



Code:

```
str(myPR)
```

Output:

```
> str(myPR)
List of 5
 $ sdev    : num [1:4] 1.708 0.956 0.383 0.144
 $ rotation: num [1:4, 1:4] 0.521 -0.269 0.58 0.565 -0.377 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
  .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
 $ center  : Named num [1:4] 5.84 3.06 3.76 1.2
  ..- attr(*, "names")= chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
 $ scale   : Named num [1:4] 0.828 0.436 1.765 0.762
  ..- attr(*, "names")= chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
 $ x       : num [1:150, 1:4] -2.26 -2.07 -2.36 -2.29 -2.38 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
 - attr(*, "class")= chr "prcomp"
>
```

Code:

myPR$x

Output:

	PC1	PC2	PC3	PC4
[1,]	-2.25714118	-0.478423832	0.127279624	0.024087508
[2,]	-2.07401302	0.671882687	0.233825517	0.102662845
[3,]	-2.35633511	0.340766425	-0.044053900	0.028282305
[4,]	-2.29170679	0.595399863	-0.090985297	-0.065735340
[5,]	-2.38186270	-0.644675659	-0.015685647	-0.035802870
[6,]	-2.06870061	-1.484205297	-0.026878250	0.006586116
[7,]	-2.43586845	-0.047485118	-0.334350297	-0.036652767
[8,]	-2.22539189	-0.222403002	0.088399352	-0.024529919
[9,]	-2.32684533	1.111603700	-0.144592465	-0.026769540
[10,]	-2.17703491	0.467447569	0.252918268	-0.039766068
[11,]	-2.15907699	-1.040205867	0.267784001	0.016675503
[12,]	-2.31836413	-0.132633999	-0.093446191	-0.133037725
[13,]	-2.21104370	0.726243183	0.230140246	0.002416941
[14,]	-2.62430902	0.958296347	-0.180192423	-0.019151375
[15,]	-2.19139921	-1.853846555	0.471322025	0.194081578
[16,]	-2.25466121	-2.677315230	-0.030424684	0.050365010
[17,]	-2.20021676	-1.478655729	0.005326251	0.188186988
[18,]	-2.18303613	-0.487206131	0.044067686	0.092779618
[19,]	-1.89223284	-1.400327567	0.373093377	0.060891973
[20,]	-2.33554476	-1.124083597	-0.132187626	-0.037630354
[21,]	-1.90793125	-0.407490576	0.419885937	0.010884821
[22,]	-2.19964383	-0.921035871	-0.159331502	0.059398340
[23,]	-2.76508142	-0.456813301	-0.331069982	0.019582826
[24,]	-1.81259716	-0.085272854	-0.034373442	0.150636353
[25,]	-2.21972701	-0.136796175	-0.117599566	-0.269238379
[26,]	-1.94532930	0.623529705	0.304620475	0.043416203
[27,]	-2.04430277	-0.241354991	-0.086075649	0.067454082
[28,]	-2.16133650	-0.525389422	0.206125707	0.010241084

```
[29,] -2.13241965 -0.312172005  0.270244895  0.083977887
[30,] -2.25769799  0.336604248 -0.068207276 -0.107918349
[31,] -2.13297647  0.502856075  0.074757996 -0.048027970
[32,] -1.82547925 -0.422280389  0.269564311  0.239069476
[33,] -2.60621687 -1.787587272 -0.047070727 -0.228470534
[34,] -2.43800983 -2.143546796  0.082392024 -0.048053409
[35,] -2.10292986  0.458665270  0.169706329  0.028926042
[36,] -2.20043723  0.205419224  0.224688852  0.168343905
[37,] -2.03831765 -0.659349230  0.482919584  0.195702902
[38,] -2.51889339 -0.590315163 -0.019370918 -0.136048774
[39,] -2.42152026  0.901161067 -0.192609402 -0.009705907
[40,] -2.16246625 -0.267981199  0.175296561  0.007023875
[41,] -2.27884081 -0.440240541 -0.034778398  0.106626042
[42,] -1.85191836  2.329610745  0.203552303  0.288896090
[43,] -2.54511203  0.477501017 -0.304745527 -0.066379077
[44,] -1.95788857 -0.470749613 -0.308567588  0.176501717
[45,] -2.12992356 -1.138415464 -0.247604064 -0.150539117
[46,] -2.06283361  0.708678586  0.063716370  0.139801160
[47,] -2.37677076 -1.116688691 -0.057026813 -0.151722682
[48,] -2.38638171  0.384957230 -0.139002234 -0.048671707
[49,] -2.22200263 -0.994627669  0.180886792 -0.014878291
[50,] -2.19647504 -0.009185585  0.152518539  0.049206884
[51,]  1.09810244 -0.860091033  0.682300393  0.034717469
[52,]  0.72889556 -0.592629362  0.093807452  0.004887251
[53,]  1.23683580 -0.614239894  0.552157058  0.009391933
[54,]  0.40612251  1.748546197  0.023024633  0.065549239
[55,]  1.07188379  0.207725147  0.396925784  0.104387166
[56,]  0.38738955  0.591302717 -0.123776885 -0.240027187
[57,]  0.74403715 -0.770438272 -0.148472007 -0.077111455
[58,] -0.48569562  1.846243998 -0.248432992 -0.040384912
[59,]  0.92480346 -0.032118478  0.594178807 -0.029779844
[60,]  0.01138804  1.030565784 -0.537100055 -0.028366154
[61,] -0.10982834  2.645211115  0.046634215  0.013714785
[62,]  0.43922201  0.063083852 -0.204389093  0.039992104
[63,]  0.56023148  1.758832129  0.763214554  0.045578465
[64,]  0.71715934  0.185602819  0.068429700 -0.164256922
[65,] -0.03324333  0.437537419 -0.194282030  0.108684396
[66,]  0.87248429 -0.507364239  0.501830204  0.104593326
[67,]  0.34908221  0.195656268 -0.489234095 -0.190869932
[68,]  0.15827980  0.789451008  0.301028700 -0.204612265
[69,]  1.22100316  1.616827281  0.480693656  0.225145511
[70,]  0.16436725  1.298259939  0.172260719 -0.051554138
[71,]  0.73521959 -0.395247446 -0.614467782 -0.083006045
[72,]  0.47469691  0.415926887  0.264067576  0.113189079
[73,]  1.23005729  0.930209441  0.367182178 -0.009911322
[74,]  0.63074514  0.414997441  0.290921638 -0.273304557
[75,]  0.70031506  0.063200094  0.444537765  0.043313222
[76,]  0.87135454 -0.249956017  0.471001057  0.101376117
[77,]  1.25231375  0.076998069  0.724727099  0.039556002
[78,]  1.35386953 -0.330205463  0.259955701  0.066604931
```

```
 [79,]  0.66258066  0.225173502 -0.085577197 -0.036318171
 [80,] -0.04012419  1.055183583  0.318506304  0.064571834
 [81,]  0.13035846  1.557055553  0.149482697 -0.009371129
 [82,]  0.02337438  1.567225244  0.240745761 -0.032663020
 [83,]  0.24073180  0.774661195  0.150707074  0.023572390
 [84,]  1.05755171  0.631726901 -0.104959762 -0.183354200
 [85,]  0.22323093  0.286812663 -0.663028512 -0.253977520
 [86,]  0.42770626 -0.842758920 -0.449129446 -0.109308985
 [87,]  1.04522645 -0.520308714  0.394464890  0.037084781
 [88,]  1.04104379  1.378371048  0.685997804  0.136378719
 [89,]  0.06935597  0.218770433 -0.290605718 -0.146653279
 [90,]  0.28253073  1.324886147 -0.089111491  0.008876070
 [91,]  0.27814596  1.116288852 -0.094172116 -0.269753497
 [92,]  0.62248441 -0.024839814  0.020412763 -0.147193289
 [93,]  0.33540673  0.985103828  0.198724011  0.006508757
 [94,] -0.36097409  2.012495825 -0.105467721  0.019505467
 [95,]  0.28762268  0.852873116 -0.130452657 -0.107043742
 [96,]  0.09105561  0.180587142 -0.128547696 -0.229191812
 [97,]  0.22695654  0.383634868 -0.155691572 -0.132163118
 [98,]  0.57446378  0.154356489  0.270743347 -0.019794366
 [99,] -0.44617230  1.538637456 -0.189765199  0.199278855
[100,]  0.25587339  0.596852285 -0.091572385 -0.058426315
[101,]  1.83841002 -0.867515056 -1.002044077 -0.049085303
[102,]  1.15401555  0.696536401 -0.528389994 -0.040385459
[103,]  2.19790361 -0.560133976  0.202236658  0.058986583
[104,]  1.43534213  0.046830701 -0.163083761 -0.234982858
[105,]  1.86157577 -0.294059697 -0.394307408 -0.016243853
[106,]  2.74268509 -0.797736709  0.580364827 -0.101045973
[107,]  0.36579225  1.556289178 -0.983598122 -0.132679346
[108,]  2.29475181 -0.418663020  0.649530452 -0.237246445
[109,]  1.99998633  0.709063226  0.392675073 -0.086221779
[110,]  2.25223216 -1.914596301 -0.396224508  0.104488870
[111,]  1.35962064 -0.690443405 -0.283661780  0.107500284
[112,]  1.59732747  0.420292431 -0.023108991  0.058136869
[113,]  1.87761053 -0.417849815 -0.026250468  0.145926073
[114,]  1.25590769  1.158379741 -0.578311891  0.098826244
[115,]  1.46274487  0.440794883 -1.000517746  0.274738504
[116,]  1.58476820 -0.673986887 -0.636297054  0.191222383
[117,]  1.46651849 -0.254768327 -0.037306280 -0.154811637
[118,]  2.41822770 -2.548124795  0.127454475 -0.272892966
[119,]  3.29964148 -0.017721580  0.700957033  0.045037725
[120,]  1.25954707  1.701046715  0.266643612 -0.064963167
[121,]  2.03091256 -0.907427443 -0.234015510  0.167390481
[122,]  0.97471535  0.569855257 -0.825362161  0.027662914
[123,]  2.88797650 -0.412259950  0.854558973 -0.126911337
[124,]  1.32878064  0.480202496  0.005410239  0.139491837
[125,]  1.69505530 -1.010536476 -0.297454114 -0.061437911
[126,]  1.94780139 -1.004412720  0.418582432 -0.217609339
[127,]  1.17118007  0.315338060 -0.129503907  0.125001677
[128,]  1.01754169 -0.064131184 -0.336588365 -0.008625505
```

```
[129,]  1.78237879  0.186735633 -0.269754304  0.030983849
[130,]  1.85742501 -0.560413289  0.713244682 -0.207519953
[131,]  2.42782030 -0.258418706  0.725386035 -0.017863520
[132,]  2.29723178 -2.617554417  0.491826144 -0.210968943
[133,]  1.85648383  0.177953334 -0.352966242  0.099675959
[134,]  1.11042770  0.291944582  0.182875741 -0.185721512
[135,]  1.19845835  0.808606364  0.164173760 -0.487849130
[136,]  2.78942561 -0.853942542  0.541093785  0.294893130
[137,]  1.57099294 -1.065013214 -0.942695700  0.035486875
[138,]  1.34179696 -0.421020154 -0.180271551 -0.214702016
[139,]  0.92173701 -0.017165594 -0.415434449  0.005220919
```

Code:

```
iris2 <- cbind(iris, myPR$x[, 1:2])
head(iris2)
```

Output:

```
> iris2 <- cbind(iris, myPR$x[, 1:2])
> head(iris2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species        PC1         PC2
1          5.1         3.5          1.4         0.2  setosa -2.257141 -0.4784238
2          4.9         3.0          1.4         0.2  setosa -2.074013  0.6718827
3          4.7         3.2          1.3         0.2  setosa -2.356335  0.3407664
4          4.6         3.1          1.5         0.2  setosa -2.291707  0.5953999
5          5.0         3.6          1.4         0.2  setosa -2.381863 -0.6446757
6          5.4         3.9          1.7         0.4  setosa -2.068701 -1.4842053
> 
```

Code:

```
install.packages(“pls”)
library(pls)
names(iris)
```

Output:

```
  [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
  > 
```

Code:

```
pcModel <- pcr(Sepal.Length~Species + Sepal.Width + Petal.Length + Petal.Width, ncomp =
3, data = iris, scale = T)
iris$pred <- predict(pcModel, iris, ncomp = 2)
head(iris)
```

Output:

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species     pred
1          5.1         3.5          1.4         0.2  setosa 5.025168
2          4.9         3.0          1.4         0.2  setosa 5.125999
3          4.7         3.2          1.3         0.2  setosa 5.073053
4          4.6         3.1          1.5         0.2  setosa 5.118447
5          5.0         3.6          1.4         0.2  setosa 5.005002
6          5.4         3.9          1.7         0.4  setosa 5.041960
>
```

Practical No: 4

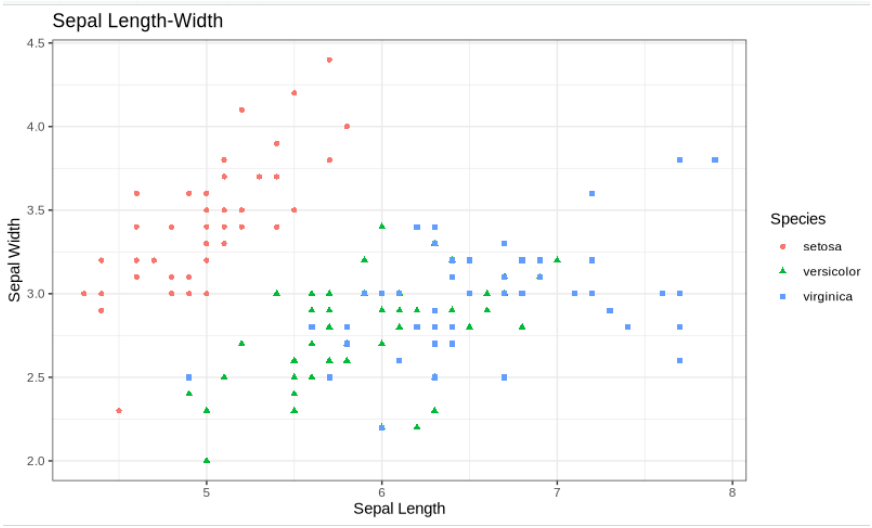
Aim: - Practical of Clustering

IRIS Data, Basic Visualization before Clustering

Code:

```
install.packages("ggplot2")
library(ggplot2)
scatter <- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))
scatter + geom_point(aes(color = Species, shape = Species)) + theme_bw() + xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Length-Width")
```

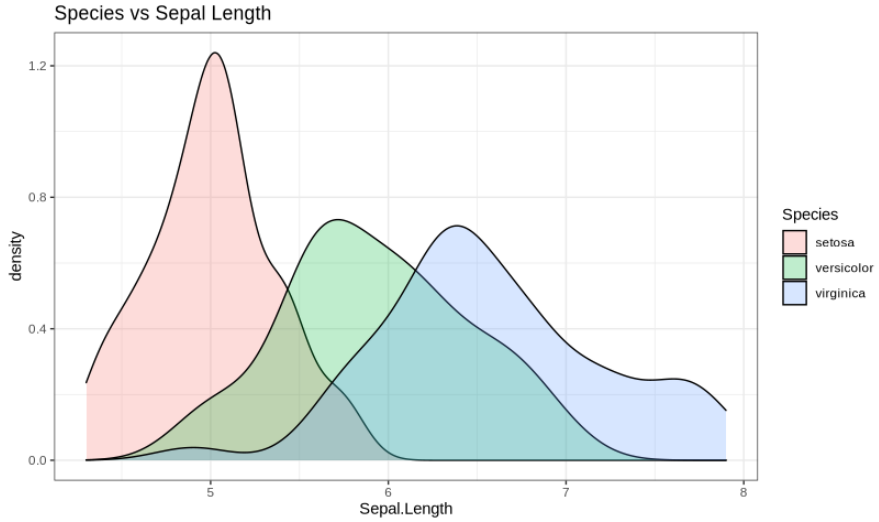
Output:



Code:

ggplot(data = iris, aes(Sepal.Length, fill = Species)) + theme_bw() + geom_density(alpha = 0.25) + labs(x = "Sepal.Length", title="Species vs Sepal Length")

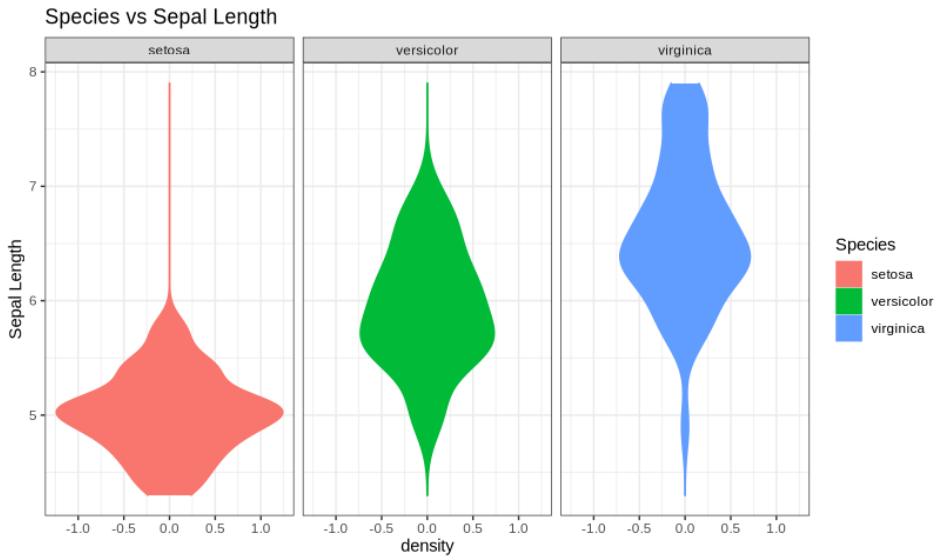
Output:



Code:

vol <- ggplot(data = iris, aes(x = Sepal.Length))
vol + stat_density(aes(ymax = ..density.., ymin = -..density.., fill = Species, color = Species), geom = "ribbon", position = "identity") + facet_grid(.~Species) + coord_flip() + theme_bw() + labs(x = "Sepal Length", title="Species vs Sepal Length")

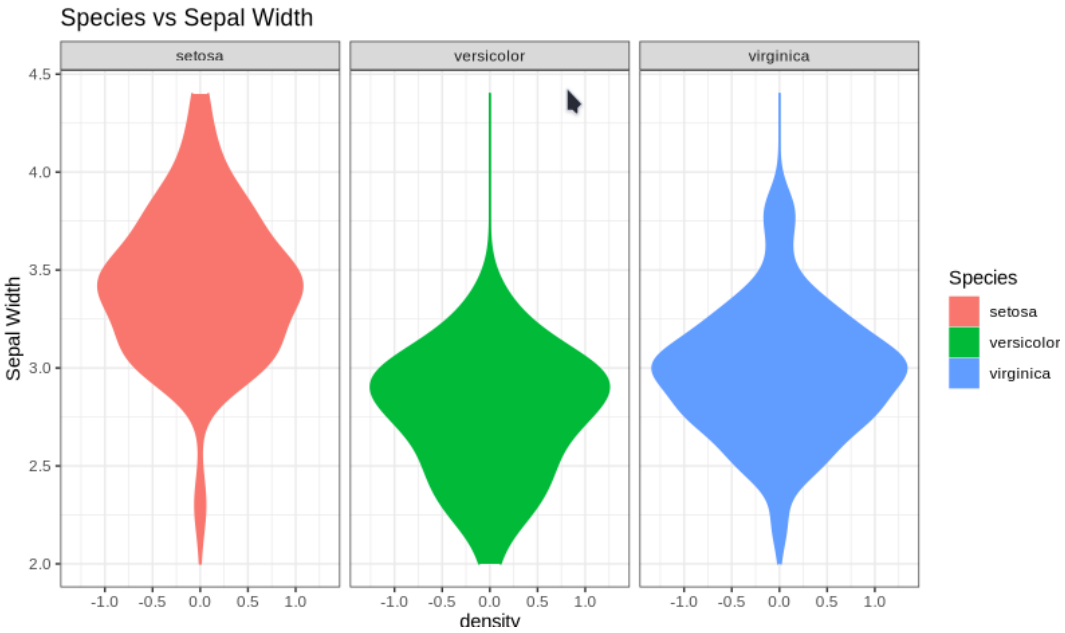
Output:



Code:

vol <- ggplot(data = iris, aes(x = Sepal.Width))
vol + stat_density(aes(ymax = ..density.., ymin = -..density.., fill = Species, color = Species), geom = "ribbon", position = "identity") + facet_grid(.~Species) + coord_flip() + theme_bw() + labs(x = "Sepal Width", title="Species vs Sepal Width")

Output:

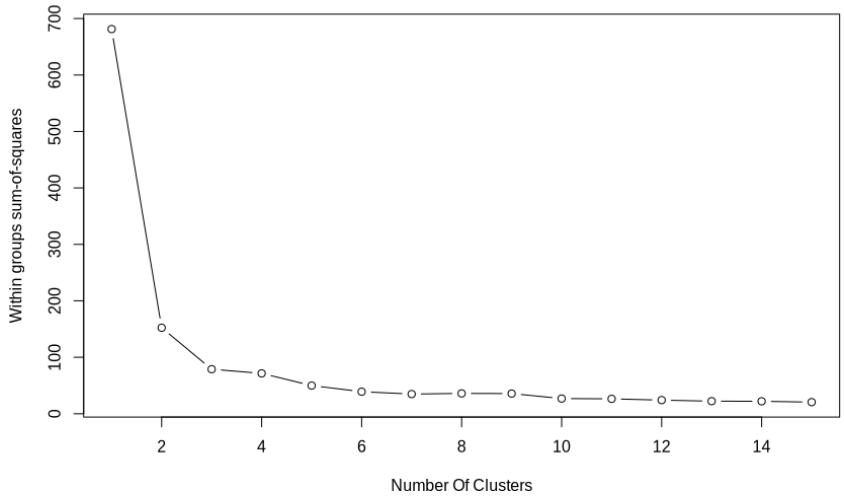


Clustering Data::Method 1

Code:

```
irisData <- iris[, 1:4]
totalWSS <- c()
for (i in 1:15) {
   clusterIRIS <- kmeans(irisData, centers = i)
   totalWSS[i] <- clusterIRIS$tot.withinss
}
plot(x = 1:15, y = totalWSS, type = "b", xlab = "Number Of Clusters", ylab = "Within groups sum-of-squares")
```

Output:

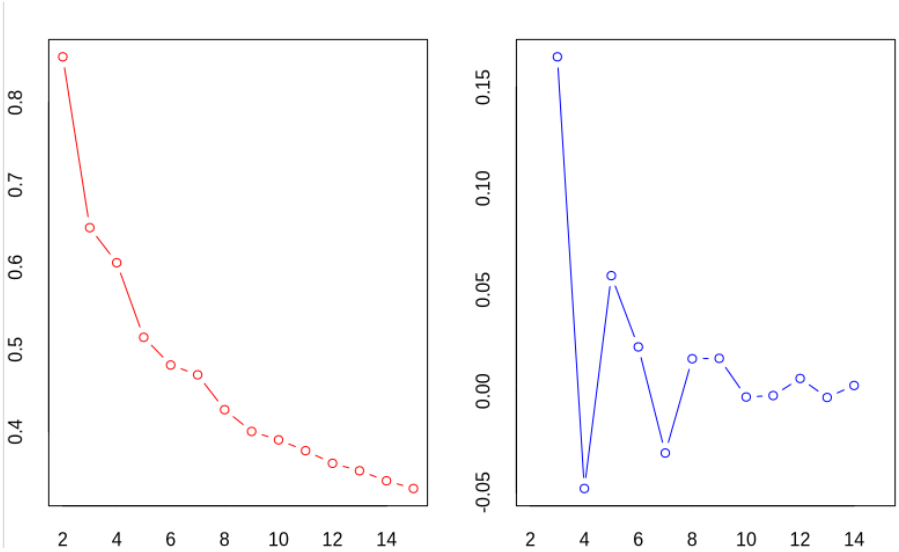


Clustering Data::Method 2
Using NbClust - Uses huge no of cluster suitability measuring critera

Code:

```
install.packages("NbClust")
library(NbClust)
par(mar = c(2, 2, 2, 2))
nb <- NbClust(irisData, method = "kmeans")
```

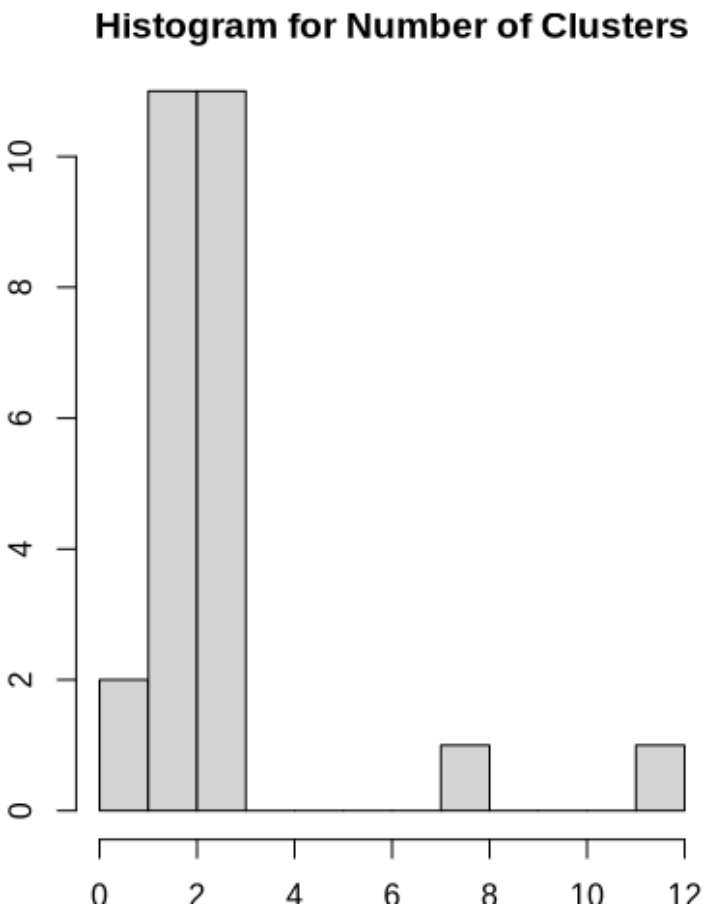
Output:



Code:

```
hist(nb$Best.nc[1, ], breaks = 15, main = "Histogram for Number of Clusters")
```

Output:

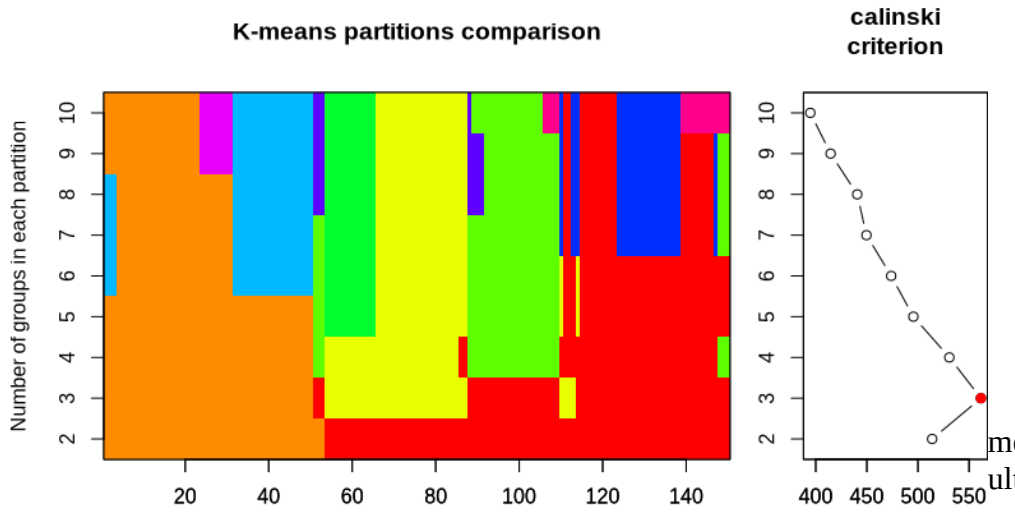


Clustering Data::Method 3

Code:

```
install.packages("vegan")
library(vegan)
modelData <- cascadeKM(irisData, 1, 10, iter = 100)
plot(modelData, sortg = TRUE)
```

Output:




```
> modelData$results[2, ]
 1 groups  2 groups  3 groups  4 groups  5 groups  6 groups  7 groups  8 groups  9 groups 10 groups
       NA  513.9245  561.6278  530.7658  495.5415  473.8506  449.6410  440.6205  414.5753  394.7207
> |
```

modelData$results[2,]

which.max(modelData$results[2,])

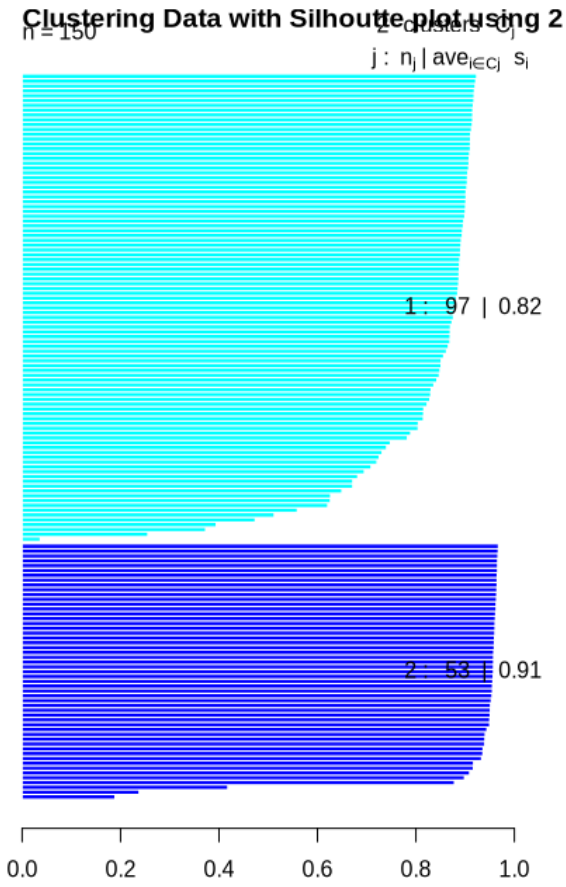
```
> which.max(modelData$results[2, ])
3 groups
       3
> |
```

Clustering Data with Silhoutte plot::Method 4

Code:

```
library(cluster)
cl <- kmeans(iris[, -5], 2)
dis <- dist(iris[, -5]) ^ 2
sil = silhouette(cl$cluster, dis)
plot(sil, main = "Clustering Data with Silhoutte plot using 2 Clusters", col = c("cyan", "blue"))
```

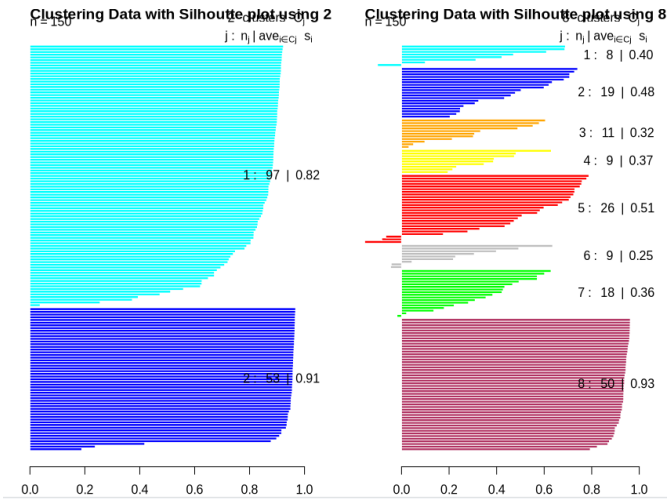
Output:



Code:

```
library(cluster)
cl <- kmeans(iris[, -5], 8)
dis <- dist(iris[, -5]) ^ 2
sil = silhouette(cl$cluster, dis)
plot(sil, main = "Clustering Data with Silhoutte plot using 8 Clusters", col = c("cyan", "blue", "orange", "yellow", "red", "gray", "green", "maroon"))
```

Output:



Code:

```
install.packages("factoextra")
install.packages("clustertend")
library(factoextra)
library(clustertend)
genx <-  function(x) {
        runif(length(x), min(x), (max(x)))
}
randomDf <- apply(iris[, -5], 2, genx)
randomDf <- as.data.frame(randomDf)
iris[, -5] <-  scale(iris[, -5])
randomDf <-  scale(randomDf)
res <- get_clust_tendency(iris[, -5], n = nrow(iris) - 1, graph = FALSE)
res$hopkins_stat
```

```
> res$hopkins_stat
[1] 0.8184781
```

```
hopkins(iris[, -5], n = nrow(iris) - 1)
```

```
> hopkins(iris[, -5], n = nrow(iris) - 1)
$H
[1] 0.1903924
```

```
res <- get_clust_tendency(randomDf, n = nrow(randomDf) - 1, graph = FALSE)
res$hopkins_stat
```

```
> res$hopkins_stat
[1] 0.5123884
```

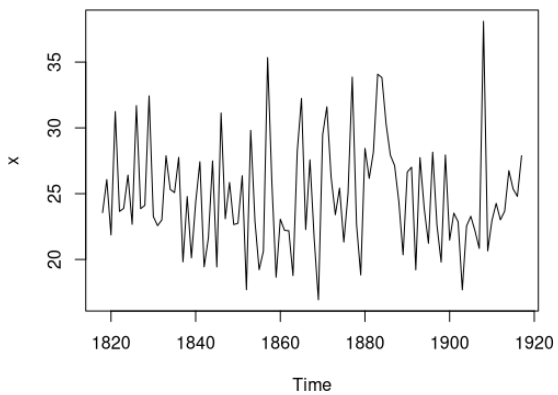
Practical No: 5

Aim: - Practical of Time-Series Forecasting

Code:

```
setwd("~/Documents/DS")
rain = read.csv("rain.csv")
rainTs = ts(rain, start = c(1818))
plot(rainTs)
```

Output:



Code:

```
rainForecast = HoltWinters(rainTs, beta = F, gamma = F)
rainForecast
```

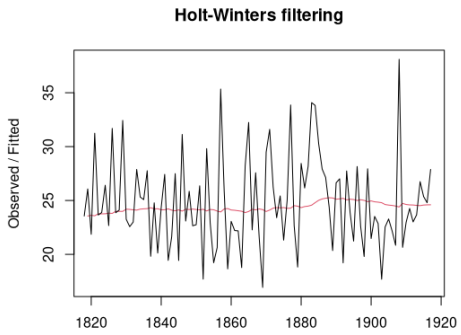
Output:

```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = rainTs, beta = F, gamma = F)

Smoothing parameters:
 alpha: 0.02412151
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 24.67819
> 
```

Code:

```
plot(rainForecast)
```

Output:



Code:

```
names(rainForecast)
```

Output:

```
[1] "fitted"       "x"            "alpha"        "beta"
[5] "gamma"        "coefficients" "seasonal"     "SSE"
[9] "call"
> 
```

Code:

```
rainForecast$fitted
```

Output:

Time Series:
Start = 1819
End = 1917
Frequency = 1
xhat level
1819 23.56000 23.56000
1820 23.62054 23.62054
1821 23.57808 23.57808
1822 23.76290 23.76290
1823 23.76017 23.76017
1824 23.76306 23.76306

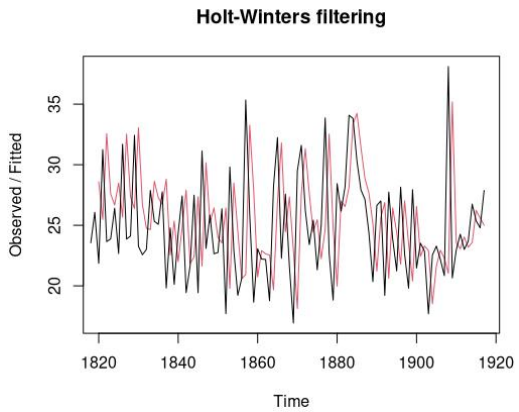
1825 23.82691 23.82691
1826 23.79900 23.79900
1827 23.98935 23.98935
1828 23.98623 23.98623
1829 23.98921 23.98921
1830 24.19282 24.19282
1831 24.17032 24.17032
1832 24.13171 24.13171
1833 24.10442 24.10442
1834 24.19549 24.19549
1835 24.22261 24.22261
1836 24.24329 24.24329
1837 24.32812 24.32812
1838 24.21938 24.21938
1839 24.23290 24.23290
1840 24.13369 24.13369
1841 24.13867 24.13867
1842 24.21782 24.21782
1843 24.10257 24.10257
1844 24.04293 24.04293
1845 24.12608 24.12608
1846 24.01280 24.01280
1847 24.18448 24.18448
1848 24.15808 24.15808
1849 24.19889 24.19889
1850 24.16153 24.16153
1851 24.12748 24.12748
1852 24.18133 24.18133
1853 24.02499 24.02499
1854 24.16454 24.16454
1855 24.13476 24.13476
1856 24.01621 24.01621
1857 23.93453 23.93453
1858 24.20964 24.20964
1859 24.25018 24.25018
1860 24.11509 24.11509
1861 24.08964 24.08964
1862 24.04430 24.04430
1863 23.99933 23.99933
1864 23.87319 23.87319
1865 23.97780 23.97780
1866 24.17710 24.17710
1867 24.13110 24.13110
1868 24.21405 24.21405
1869 24.15075 24.15075
1870 23.97658 23.97658
1871 24.10933 24.10933
1872 24.29001 24.29001

1873 24.33729 24.33729
1874 24.31468 24.31468
1875 24.34134 24.34134
1876 24.26847 24.26847
1877 24.28659 24.28659
1878 24.51752 24.51752
1879 24.47295 24.47295
1880 24.33660 24.33660
1881 24.43558 24.43558
1882 24.47717 24.47717
1883 24.56625 24.56625
1884 24.79573 24.79573
1885 25.01341 25.01341
1886 25.14045 25.14045
1887 25.20750 25.20750
1888 25.25411 25.25411
1889 25.23351 25.23351
1890 25.11571 25.11571
1891 25.15248 25.15248
1892 25.19729 25.19729
1893 25.05286 25.05286
1894 25.11768 25.11768
1895 25.08710 25.08710
1896 24.99407 24.99407
1897 25.07019 25.07019
1898 25.01085 25.01085
1899 24.88515 24.88515
1900 24.95884 24.95884
1901 24.87469 24.87469
1902 24.84201 24.84201
1903 24.79420 24.79420
1904 24.62284 24.62284

Code:

```
r2 = HoltWinters(rainTs, alpha = 0.8, gamma = F)
plot(r2)
```

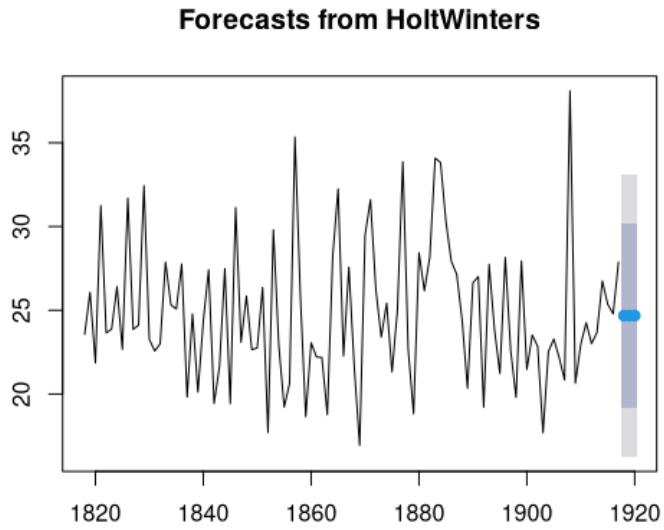
Output:



Code:

```
rf = forecast::forecast(rainForecast, h = 3)
plot(rf)
```

Output:



Code:

```
rf
```

Output:

```
> rf
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
1918       24.67819 19.17493 30.18145 16.26169 33.09470
1919       24.67819 19.17333 30.18305 16.25924 33.09715
1920       24.67819 19.17173 30.18465 16.25679 33.09960
>
```

Practical No:

6

Aim:Practical of Simple/Multiple Linear Regression

Code:

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

Output:

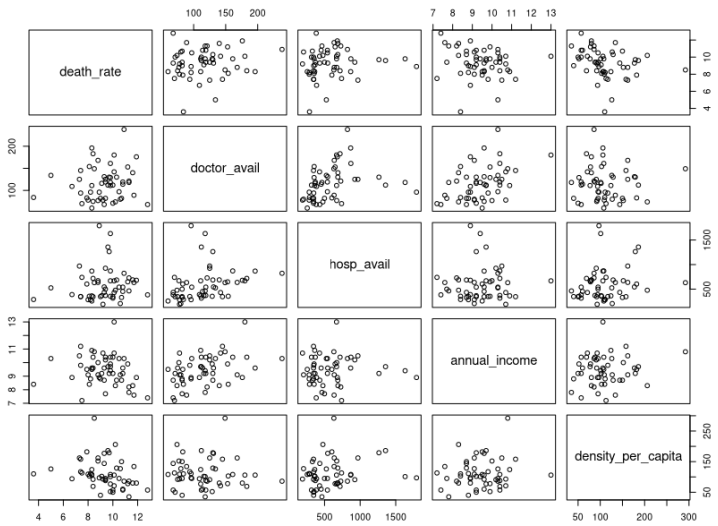
```
     price            bedrooms        bathrooms      sqft_living       sqft_lot          floors        condition
 Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290   Min.   :    520   Min.   :1.000   Min.   :1.000
 1st Qu.: 321950   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1427   1st Qu.:   5040   1st Qu.:1.000   1st Qu.:3.000
 Median : 450000   Median : 3.000   Median :2.250   Median : 1910   Median :   7618   Median :1.500   Median :3.000
 Mean   : 540182   Mean   : 3.371   Mean   :2.115   Mean   : 2080   Mean   :  15107   Mean   :1.494   Mean   :3.409
 3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10688   3rd Qu.:2.000   3rd Qu.:4.000
 Max.   :7700000   Max.   :33.000   Max.   :8.000   Max.   :13540   Max.   :1651359   Max.   :3.500   Max.   :5.000
     grade          sqft_above    sqft_basement
 Min.   : 1.000   Min.   : 290   Min.   :   0.0
 1st Qu.: 7.000   1st Qu.:1190   1st Qu.:   0.0
 Median : 7.000   Median :1560   Median :   0.0
 Mean   : 7.657   Mean   :1788   Mean   : 291.5
 3rd Qu.: 8.000   3rd Qu.:2210   3rd Qu.: 560.0
 Max.   :13.000   Max.   :9410   Max.   :4820.0
>
```

Code:
names(house)

Output:
```
> names(house)
 [1] "price"         "bedrooms"      "bathrooms"     "sqft_living"   "sqft_lot"      "floors"        "condition"
 [8] "grade"         "sqft_above"    "sqft_basement"
>
```

Code:
pairs(~death_rate+doctor_avail+hosp_avail+annual_income+density_per_capita,data = house)

Output:



Code:
housemodel <-
lm(density_per_capita~death_rate+doctor_avail+hosp_avail+annual_income,data=house)
housemodel

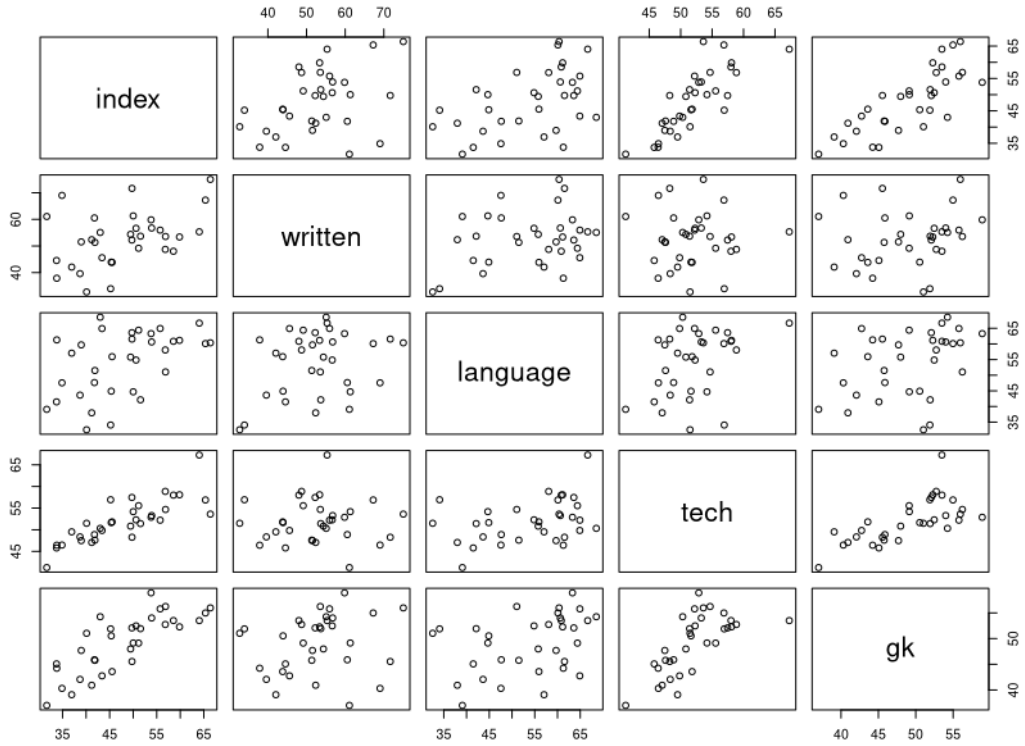
Output:
```
Call:
lm(formula = density_per_capita ~ death_rate + doctor_avail +
    hosp_avail + annual_income, data = house)

Coefficients:
  (Intercept)     death_rate   doctor_avail     hosp_avail  annual_income
    125.88097       -7.65709       -0.14499        0.03509        5.52670

>
```

Code:
index<-read.csv(file.choose(),sep=",",header = T)
names(index)

Output:

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

Code:
pairs(~index+written+language+tech+gk,data = index)
Output:



Code:
model1<-lm(index~.,data = index)
summary(model1)
Output:

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

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

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

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

Code:

```
index$pred<-fitted(model1)
head(index)
```

Output:

```
  empid index written language  tech    gk     pred
1     1 45.52   43.83    55.92 51.82 43.58 44.02220
2     2 40.10   32.71    32.56 51.49 51.03 42.55279
3     3 50.61   56.64    54.84 52.29 52.47 53.12213
4     4 38.97   51.53    59.69 47.48 47.69 43.41803
5     5 41.87   51.35    51.50 47.59 45.77 41.97188
6     6 38.71   39.60    43.63 48.34 42.06 36.52202
```

Code:

```
index$res<-residuals(model1)
head(index)
```

Output:

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

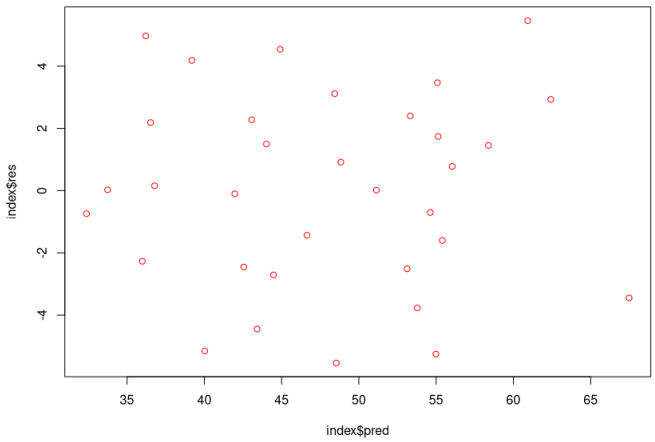
Code:

```
install.packages(car)
library(car)
vif(model1)
```

Output:

```
    empid  written language     tech       gk
1.119815 1.185225 1.344122 2.178955 2.033284
```

Code:
plot(index$pred,index$res,col="red")
Output:



Code:
shapiro.test(index$res)
Output:

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

        Shapiro-Wilk normality test

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

Code:
ncvTest(model1,~written+language+tech+ gk)
Output:

```
Non-constant Variance Score Test
Variance formula: ~ written + language + tech + gk
Chisquare = 2.146914, Df = 4, p = 0.70876
```

Code:
durbinWatsonTest(model1)
Output:

```
> durbinWatsonTest(model1)
 lag Autocorrelation D-W Statistic p-value
   1       0.2268414      1.500571   0.102
 Alternative hypothesis: rho != 0
>
```

Code:

```
library(caret)
library(lattice)
library(ggplot2)
index<-read.csv(file.choose(),sep=",",header = T)
```

summary(index)

Output:

```
> summary(index)
     empid        index          written         language          tech             gk
 Min.   : 1   Min.   :31.64   Min.   :32.71   Min.   :32.56   Min.   :41.25   Min.   :37.00
 1st Qu.: 9   1st Qu.:41.19   1st Qu.:45.59   1st Qu.:44.89   1st Qu.:48.34   1st Qu.:45.07
 Median :17   Median :49.45   Median :53.38   Median :57.04   Median :51.64   Median :50.53
 Mean   :17   Mean   :47.87   Mean   :52.66   Mean   :53.99   Mean   :52.02   Mean   :49.04
 3rd Qu.:25   3rd Qu.:53.92   3rd Qu.:56.75   3rd Qu.:61.28   3rd Qu.:54.68   3rd Qu.:53.50
 Max.   :33   Max.   :66.39   Max.   :75.03   Max.   :68.53   Max.   :67.27   Max.   :58.90
>
```

Code:

data<-createDataPartition(index$empid,p=0.8,list=F)
head(data)

Output:

```
> head(data)
     Resample1
[1,]         1
[2,]         2
[3,]         3
[4,]         4
[5,]         5
[6,]         6
>
```

Code:

dim(data)

Output:

```
> dim(data)
[1] 29  1
>
```

Code:

traindata<-index[data,]
dim(traindata)

Output:

```
> dim(traindata)
[1] 29  6
```

Code:

testdata<-index[-data,]
dim(testdata)

Output:

```
> dim(testdata)
[1] 4 6
```

Code:

names(traindata)

Output:

```
> names(traindata)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
>
```

Code:

```
modeltrain<-lm(index~written+language+tech+gk,data=traindata)
modeltrain$res<-residuals(modeltrain)
RMSEtrain<-sqrt(mean(modeltrain$res**2))
RMSEtrain
```

Output:

```
[1] 2.829915
```

Code:

```
testdata$pred<-predict(modeltrain,testdata)
testdata$res<-testdata$index-testdata$pred
RMSEtest<-sqrt(mean(testdata$res**2))
RMSEtest
```

Output:

```
[1] 5.952287
```

Code:

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

Output:

```
Linear Regression

33 samples
 4 predictor

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

  RMSE      Rsquared   MAE
  3.788449  0.8393999  3.074181

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

Code:

```
kfoldsrp<-trainControl(method = "repeatedcv",number = 4,repeats = 5)
modelkfoldsrp<-train(index~written+language+tech+gk,data =
index,method="lm",trControl=kfoldsrp)
modelkfoldsrp
```

Output:

```
Linear Regression

33 samples
 4 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 25, 25, 25, 24, 24, 25, ...
Resampling results:

  RMSE      Rsquared   MAE
  3.822284  0.8740461  3.162349

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

Code:

```
kfoldsloocv<-trainControl(method = "LOOCV")
kfoldsloocvmodel<-train(index~written+language+tech+gk,data =
index,method="lm",trControl=kfoldsloocv)
kfoldsloocvmodel
```

Output:

```
Linear Regression

33 samples
 4 predictor

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 32, 32, 32, 32, 32, 32, ...
Resampling results:

  RMSE      Rsquared   MAE
  4.044207  0.8147009  3.254919

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

Code:

```
null<-lm(index~1,data=index)
full<-lm(index~.,data = index)
names(index)
```

Output:

```
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
```

Code:

```
step(null,scope = list(lower=null,upper=full),direction = "forward")
```

Output:

```
Start:  AIC=149.28
index ~ 1

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

```
+ empid     1     62.42 2800.31 150.55

Step:  AIC=116.4
index ~ tech

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

Step:  AIC=96
index ~ tech + written

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

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

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

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

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

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

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

Code:

step(full,scope=list(lower=null,upper=full),direction = "backward")

Output:

```
Start:  AIC=85.79
index ~ empid + written + language + tech + gk

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

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

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

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

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

Practical No: 7

Aim: Practical of Logistics Regression

Code:
```
library(datasets)
ir_data<- iris
head(ir_data)
```
Output:
```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1   -0.8976739  1.01560199    -1.335752   -1.311052  setosa
2   -1.1392005 -0.13153881    -1.335752   -1.311052  setosa
3   -1.3807271  0.32731751    -1.392399   -1.311052  setosa
4   -1.5014904  0.09788935    -1.279104   -1.311052  setosa
5   -1.0184372  1.24503015    -1.335752   -1.311052  setosa
6   -0.5353840  1.93331463    -1.165809   -1.048667  setosa
```

Code:
```
str(ir_data)
```
Output:
```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  -0.898 -1.139 -1.381 -1.501 -1.018 ...
 $ Sepal.Width : num  1.0156 -0.1315 0.3273 0.0979 1.245 ...
 $ Petal.Length: num  -1.34 -1.34 -1.39 -1.28 -1.34 ...
 $ Petal.Width : num  -1.31 -1.31 -1.31 -1.31 -1.31 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
>
```

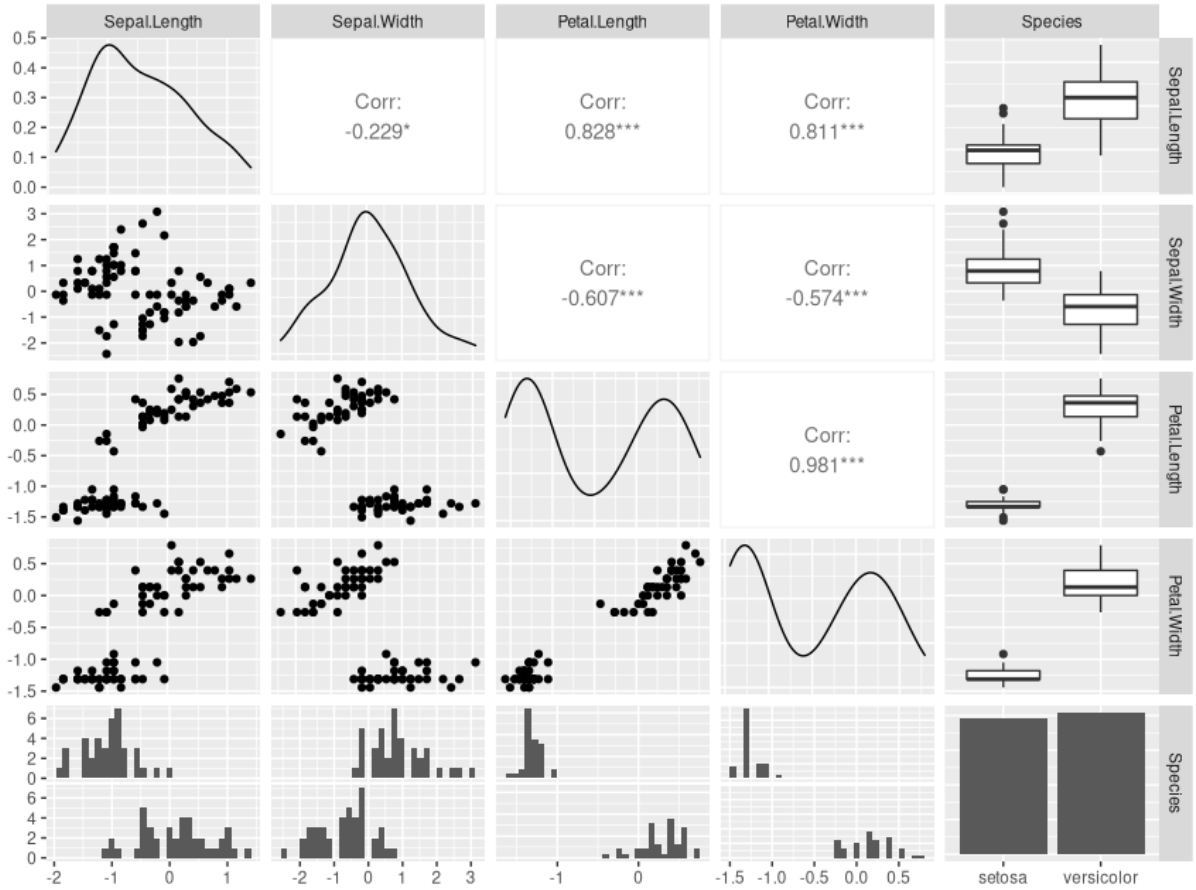
Code:
```
levels(ir_data$Species)
```
Output:
```
[1] "setosa"     "versicolor" "virginica"
```

Code:
```
sum(is.na(ir_data))
```
Output:
```
[1] 0
```

Code:
```
install.packages("ggplot2")
install.packages("GGally")
ir_data<-ir_data[1:100,]
set.seed(100)
samp<-sample(1:100,80)
ir_test<-ir_data[samp,]
ir_ctrl<-ir_data[-samp,]
library(ggplot2)
library(GGally)
ggpairs(ir_test)
```

Output:

```
plot: [5,1]
[=========================================================================
===>---------------] 84% est: 0s `stat_bin()` using `bins = 30`. Pick
better value with `binwidth`.
 plot: [5,2]
[=========================================================================
=======>-----------] 88% est: 0s `stat_bin()` using `bins = 30`. Pick
better value with `binwidth`.
 plot: [5,3]
[=========================================================================
==========>-------] 92% est: 0s `stat_bin()` using `bins = 30`. Pick
better value with `binwidth`.
 plot: [5,4]
[=========================================================================
=============>----] 96% est: 0s `stat_bin()` using `bins = 30`. Pick
better value with `binwidth`.
```



Code:

```
y<-ir_test$Species; x<-ir_test$Sepal.Length
glfit<-glm(y~x, family = 'binomial')
summary(glfit)
```

Output:

```
Call:
glm(formula = y ~ x, family = "binomial")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.12681  -0.51865   0.02993   0.30652   2.25044

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.3732     0.6645   3.571 0.000355 ***
x             4.2333     0.9181   4.611 4.01e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 110.854  on 79  degrees of freedom
Residual deviance:  48.818  on 78  degrees of freedom
AIC: 52.818

Number of Fisher Scoring iterations: 6

> |
```

Code:

```
newdata<- data.frame(x=ir_ctrl$Sepal.Length)
predicted_val<-predict(glfit, newdata, type="response")
prediction<-data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species,predicted_val)
prediction
```

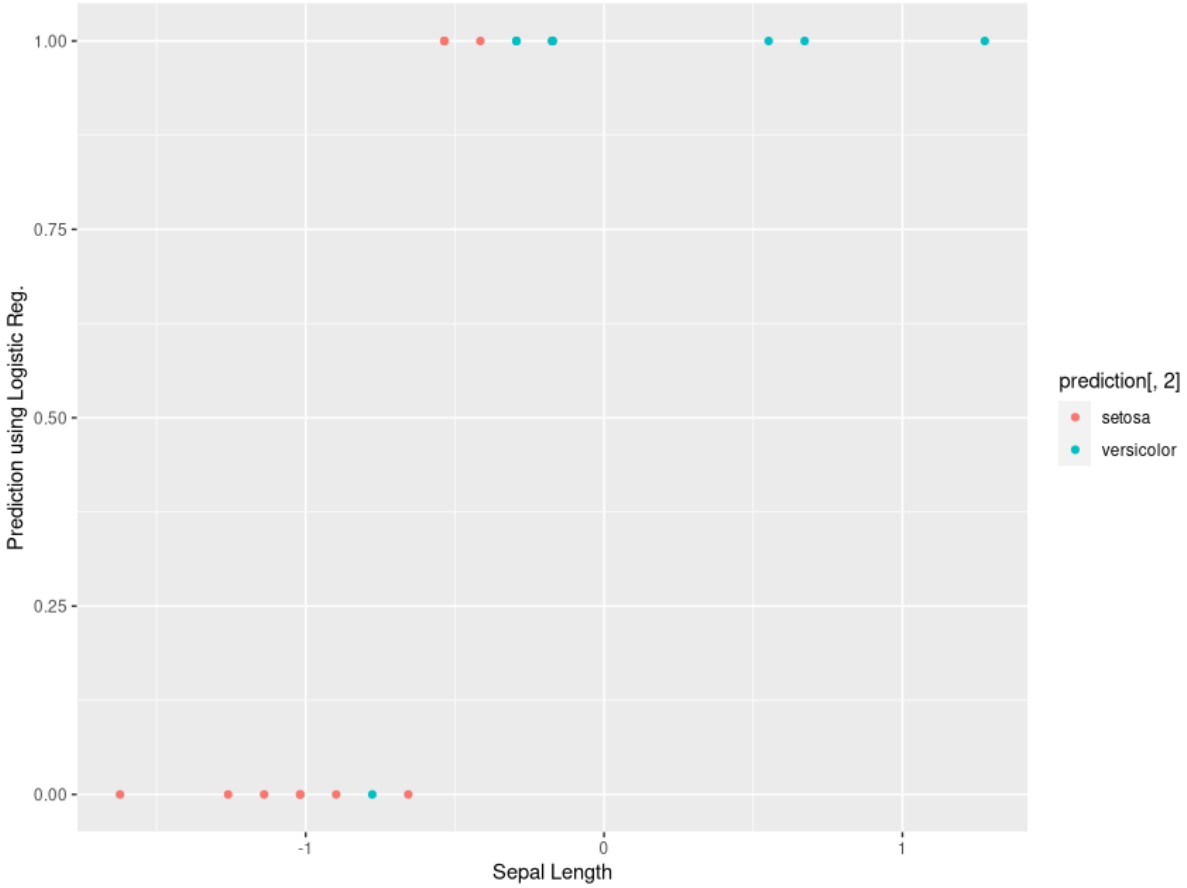
Output:

```
   ir_ctrl.Sepal.Length ir_ctrl.Species predicted_val
1            -0.5353840          setosa    0.52665832
2            -1.0184372          setosa    0.12584710
3            -1.2599638          setosa    0.04923563
4            -0.5353840          setosa    0.52665832
5            -0.1730941          setosa    0.83759291
6            -1.1392005          setosa    0.07948111
7            -0.4146207          setosa    0.64975559
8            -0.8976739          setosa    0.19357325
9            -1.6222537          setosa    0.01104861
10           -1.0184372          setosa    0.12584710
11           -0.6561473          setosa    0.40023260
12            1.2760656      versicolor    0.99958015
13           -0.1730941      versicolor    0.83759291
14           -0.7769106      versicolor    0.28582944
15           -0.2938574      versicolor    0.75569041
16           -0.2938574      versicolor    0.75569041
17            0.5514857      versicolor    0.99105619
18            0.6722490      versicolor    0.99461661
19           -0.1730941      versicolor    0.83759291
20           -0.1730941      versicolor    0.83759291
> |
```

Code:

qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab = 'Prediction using Logistic Reg.')

Output:



Practical No: 8

Aim: Practical of Hypothesis testing

1. One-sample hypothesis test

Code:

```
x= c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6, + 8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4, 10.8, +11.3, 11.9)
t.test(x-9,alternative="two.sided",conf.level=0.95)
```

Output:

```
        One Sample t-test

data:  x - 9
t = -0.35687, df = 23, p-value = 0.7244
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.7079827  0.4996494
sample estimates:
 mean of x
-0.1041667

> |
```

2. Two sample hypothesis test

Code:

```
x=c(418,421,421,422,425,427,431,434,437,439,446,447,448,453,454,463,465)
y=c(429,430,430,431,36,437,440,441,445,446,447)
test2<-t.test(x,y,alternative="two.sided",mu=0,var.equal=F,conf.level=0.95)
test2
```

Output:

```
        Welch Two Sample t-test

data:  x and y
t = 1.0123, df = 10.202, p-value = 0.3348
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -44.46343 118.86984
sample estimates:
mean of x mean of y
 438.2941  401.0909

> |
```

Practical No: 9

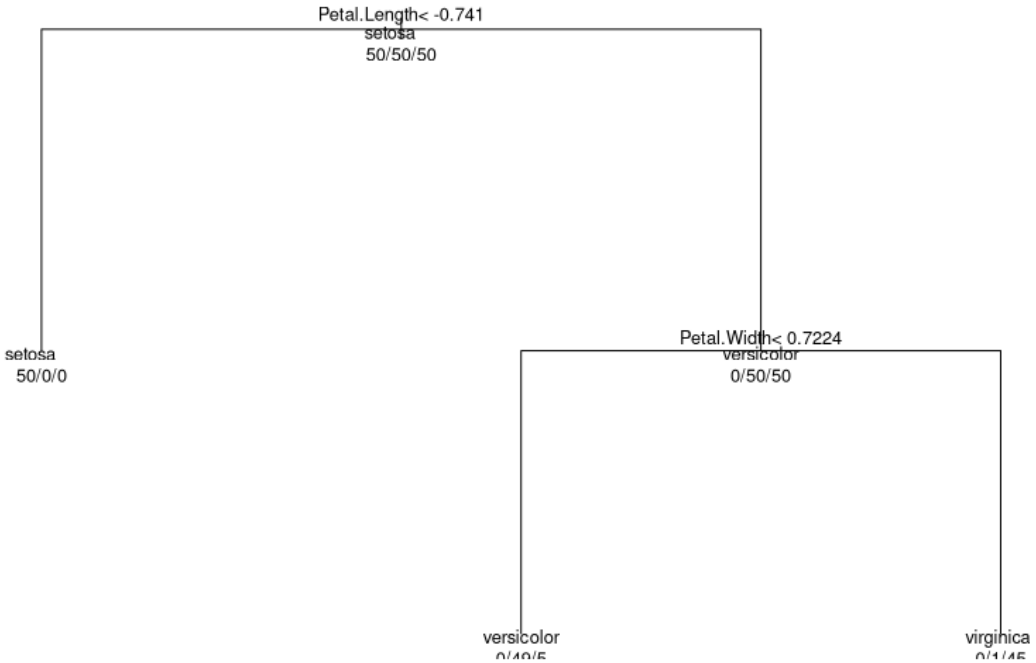
Aim: Practical of Decision Tree

Decision Tree using R:

Code:

```
install.packages("rpart")
mydata<-data.frame(iris)
attach(mydata)
library(rpart)
model<-rpart(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
data=mydata, method="class")
plot(model)
text(model,use.n=TRUE,all=TRUE,cex=0.8)
```

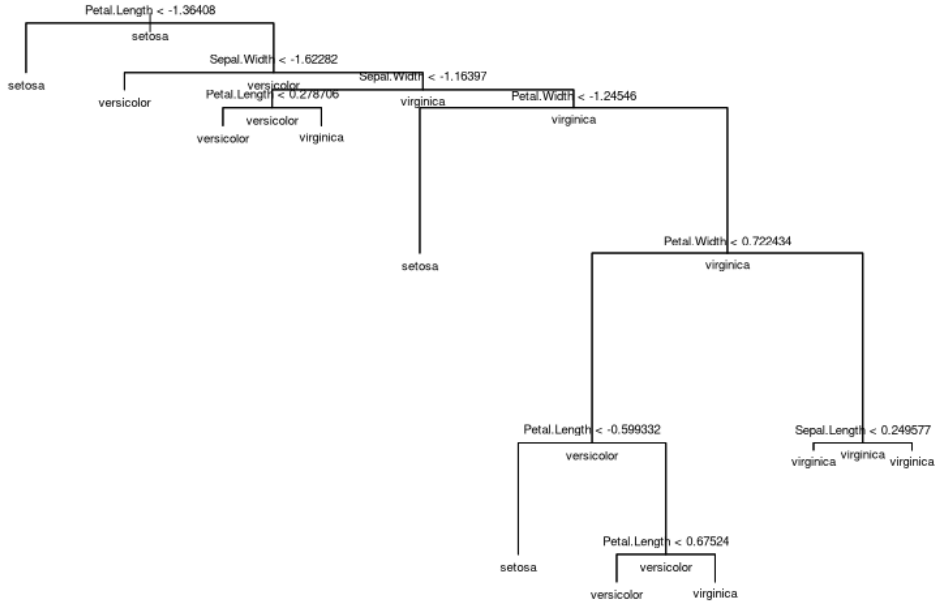
Output:



Code:

```
library(tree)
model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
data=mydata, method="class", split="gini")
plot(model1)
text(model1,all=TRUE,cex=0.6)
```

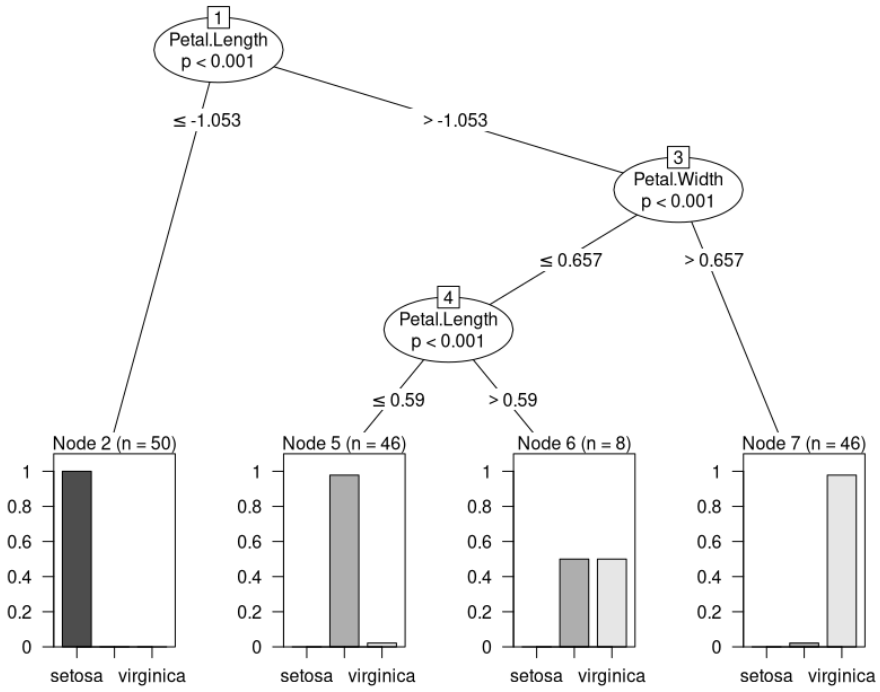
Output:



Code:

```
library(party)
model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
data=mydata)
plot(model2)
```

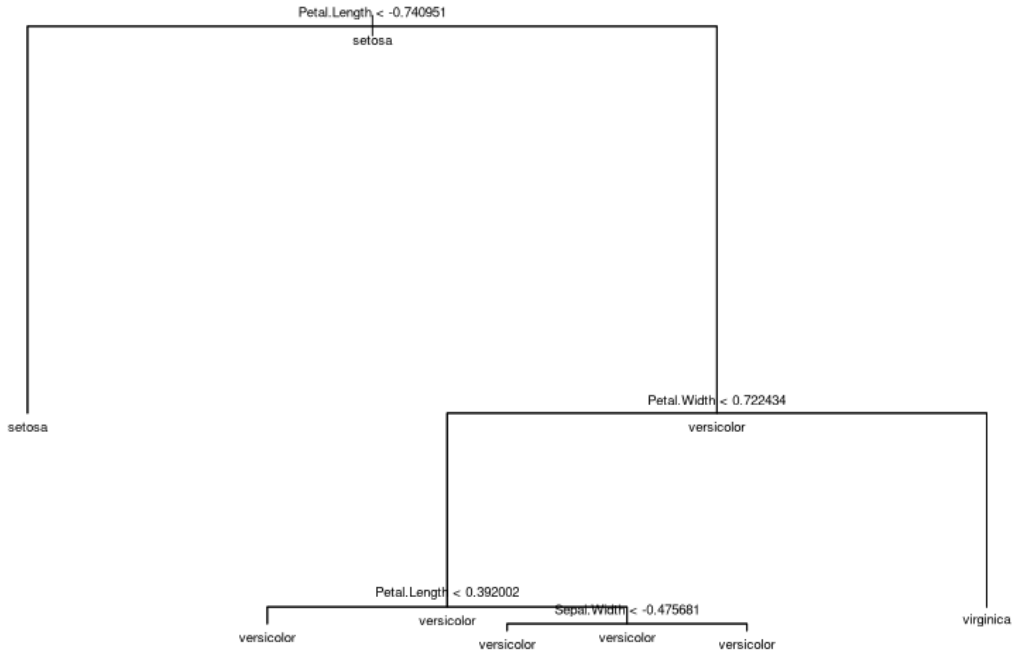
Output:



Code:

```
library(tree)
mydata<-data.frame(iris)
attach(mydata)
model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
        data=mydata,        method="class",        control =  tree.control(nobs =
150, mincut = 10))
plot(model1)
text(model1,all=TRUE,cex=0.6)
```

Output:



Code:

```
predict(model1,iris)
```

Output:

```
    setosa versicolor  virginica
1        1 0.00000000 0.00000000
2        1 0.00000000 0.00000000
3        1 0.00000000 0.00000000
4        1 0.00000000 0.00000000
5        1 0.00000000 0.00000000
6        1 0.00000000 0.00000000
7        1 0.00000000 0.00000000
8        1 0.00000000 0.00000000
9        1 0.00000000 0.00000000
10       1 0.00000000 0.00000000
11       1 0.00000000 0.00000000
12       1 0.00000000 0.00000000
13       1 0.00000000 0.00000000
14       1 0.00000000 0.00000000
```

Code:

model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = mydata, controls = ctree_control(maxdepth=2))
plot(model2)

Output: