Exam Seat No._____

THAKUR COLLEGE OF SCIENCE & COMMERCE

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

THAKUR TRUSTS

ISO 9001 : 2015 Certified

Best College Award by University of Mumbai for the Year 2018-2019

Degree College

Computer Journal
CERTIFICATE

SEMESTER 6 **UID No.** 2020790

Class TYBSc(CS) **Roll No.** 327 **Year** 2022-23

This is to certify that the work entered in this journal is the work of Mst. / Ms. Mohit Kapoor

who has worked for the year 2022-23 in the Computer Laboratory.

Teacher In-Charge

Head of Department

Date :_____

Examiner

INDEX:

327 Mohit Kapoor

Practical 1

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

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

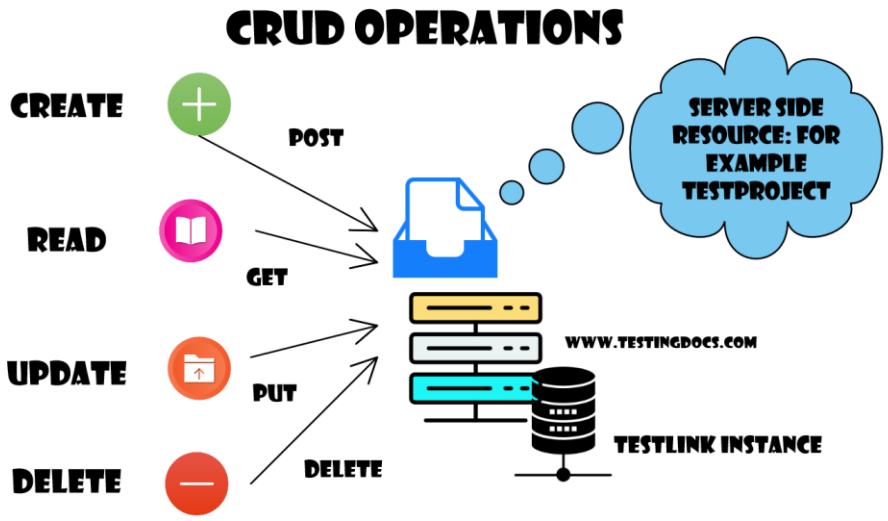
- The document model maps to the objects in your application code, making data easy to work with
- Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
- MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic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

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

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

Most applications have some form of CRUD functionality. In fact, every programmer has had to deal with CRUD at some point. Not to mention, a CRUD application is one that utilizes forms to retrieve and return data from a database.

The first reference to CRUD operations came from Haim Kilov in 1990 in an article titled, “From semantic to object-oriented data modeling.” However, the term was first made popular by James Martin’s 1983 book, *Managing the Data-base Environment*.



327 Mohit Kapoor

Code:

Starting server with mongo or mongodb

C: \ >mongo >db Test

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

```
use database_name
```

For example: create a database “tycs”:

> use tycs switched to db tycs

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

Create a collection user and insert a document in it.

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

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

>showdbs admin 0.000GB config

0.000GB local 0.000GB tycs

0.000GB

- MongoDB Drop Database

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

O/P:

327 Mohit Kapoor
("dropped" : "Testdb", "ok" : 1 } MongoDB

Enterprise > show dbs admin 0.000GB config

0.000GB local 0.000GB tycs 0.000GB O/P:

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

• Create Co1lect1ozz 1zz MozzgoDB

Method 1: Creatlng the Collection 1n MongoDB on the fly MongoDB

Enterprise > use tycs switched to db tycs

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

WriteResult(("nInserted" : 1 })

Syntax: db.co1leetion_name.find{J

MongoDB Enterprise > db.tycs.find()

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

```
show collections
```

MongoDB Enterprise > show collections

O/P: tycs user

• Drop collection Ie MongoDB

SYNTAX

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

327 Mohit Kapoor

MongoDB Enterprise > use students switched to db

students MongoDB Enterprise > show collections

students teachers tycs user

MongoDB Enterprise > db.user.drop{J true

MongoDB Enterprise > show collections students Teacher

tycs

MongoDB Insert Document

Syntax to insert a document into the collection:

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

> db.tycs.insert{

zzazzze: "ASIE'", age: 20,

email: "as1f@gmai1.com",

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

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

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

Verification:

Syntax:

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

> db.tycs.find{J

(" id" : Objectld("5c2d37734fa204bd77e7fc1c"), "name" : "ASIF", "age" : 20, "email" : "asif@gmail.com",

"course" : [{ "name" :

"MongoDB", "duration" : 7 },

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

MongoDB Example: Insert Multiple Documents in collection

MongoDB Enterprise > var beginners=

... "studentID": 1001,

... "studentName":"Asif",

... "age":20

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

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

"_id" ObjectId("5e4l0f3fe3755ble06a63d1e"),

"studentID" : 1001,

"studentName" : "Asif",

"age" : 20

• Quezy Document based on tfze criteria

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

_id" ObjectId("5c28lc90c23e08d l5l5fd9cc"),

"StudentId" : 1001,

"StudentName" : "ASlf",

"age" : 20

• Updating Document using update() method

Syntax:

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

> use tycs switched to db tycs > sbow

co11ectlozzs beginnersbook students

tycs

> db.createCol1eetion{"got"} (

"ok" : 1 }

> var abc = [

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

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

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

"name" : "steve",

"age" : 20

To update multiple documents with the update{} method

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

Updating Document using save() method Syntax:

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

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

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

"did" ObjectId("59bd2e73ce524b733fl4dd65"),

"name" : "Asifi',

"age" : 20

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

"_id" ObjectId("59bd2e73ce524b733fl4dd65"),

"name" : "Steve",

"age" : 20

• MongoDB Delete Document from a Collection

Syntax of remove(J method

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

Delete Document using remove{J method

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

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

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

Output:

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

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

It will list all the documents of students collection.

> use tycs switched to db tycs

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

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

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

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

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

"age" : 30

Removes all Documents

- ***MongoDB Projection***

Syntax:

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

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

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

"Studentld" : 1001,

327 Mohit Kapoor
"StudentName" : "Steve",

"age" : 20

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

("StudentId" : 1001 } ("StudentId" :

1002 }

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

("Studentld" : 1001 }

{ "Studentld" : 1002 }

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

Syntax:

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

]

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

MongoDB Sk1p() Method

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

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

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

The default value 1s 1

. For example: collection studentdata contains following documents:

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

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

Let's display theJS t₁ Ident_id of all the documents in descending order

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

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

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

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

• MongoDB Indexing with Example How to create index in

MongoDB

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

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

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

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

Output:

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

- ***MongoDB — Finding the indexes in a collection***

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

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

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

327 Mohit Kapoor

Practical 2

Aim: Demonstration of Simple and Linear Regression

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

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

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

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

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

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable. You can use multiple linear regression when you want to know:

How strong the relationship is between two or more independent variables and one dependent variable (e.g. how rainfall, temperature, and amount of fertilizer added affect crop growth).

The value of the dependent variable at a certain value of the independent variables (e.g. the expected yield of a crop at certain levels of rainfall, temperature, and fertilizer addition).

327 Mohit Kapoor

Code:

Simple Regression:

#Simple Linear Regression.py 9+

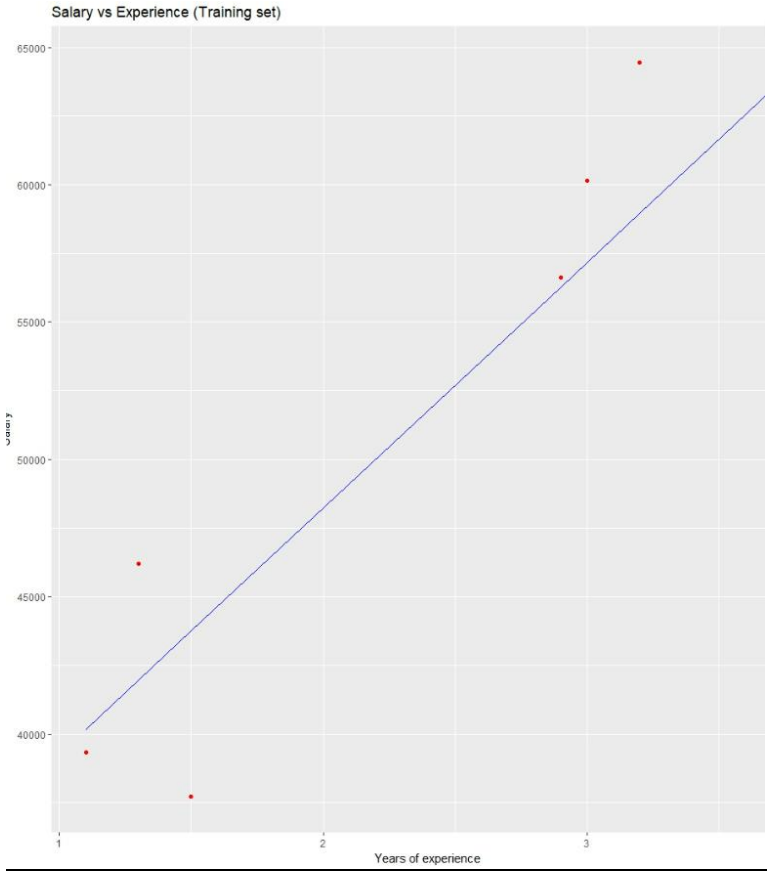
C: > Users > Mohit Kapoor > Desktop > Mohit > College > DS > #Simple Linear Regression.py > ...

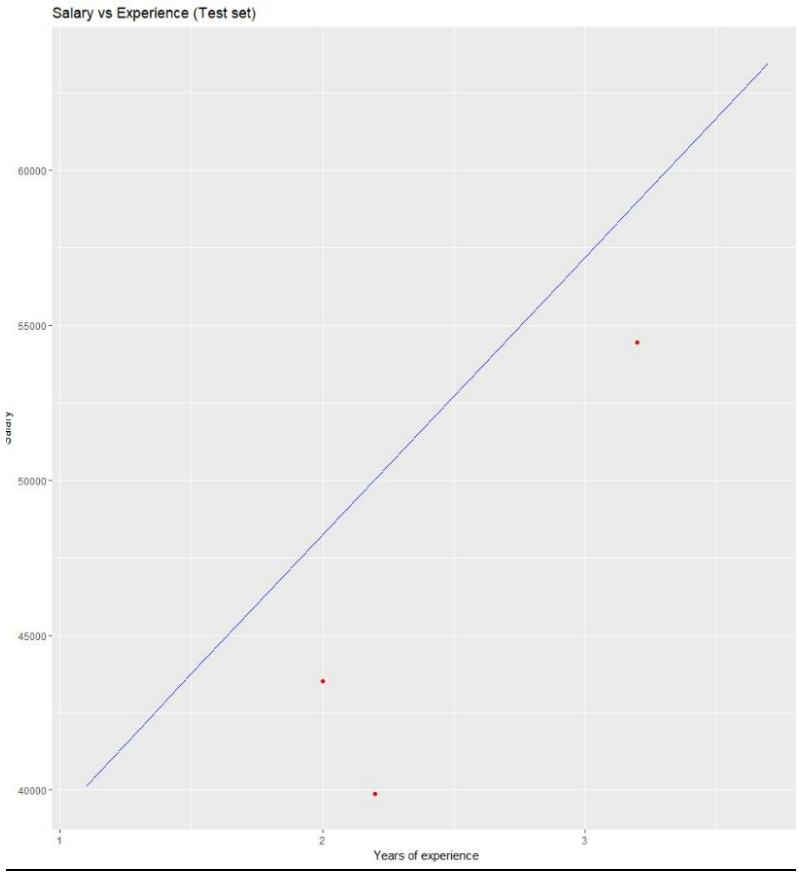
```r
# Simple Linear Regression
# Importing the dataset
dataset = read.csv('salary.csv')

# Splitting the dataset into the
# Training set and Test set
install.packages('caTools')
library(caTools)
split = sample.split(datasetSalary, SplitRatio = 0.7)
trainingset = subset(dataset, split == TRUE)
testset = subset(dataset, split == FALSE)

# Fitting Simple Linear Regression to the Training set
lm.r= lm(formula = Salary,YearsExperience,
                    data = trainingset)
coef(lm.r)

# Predicting the Test set results
ypred = predict(lm.r, newdata = testset)

install.packages("ggplot2")
library(ggplot2)

# Visualising the Training set results
ggplot() + geom_point(aes(x = trainingset$YearsExperience,
                          y = trainingset$Salary), colour = 'red') +
geom_line(aes(x = trainingset$YearsExperience,
y = predict(lm.r, newdata = trainingset)), colour = 'blue') +

ggtitle('Salary vs Experience (Training set)') +
xlab('Years of experience') +
ylab('Salary')

# Visualising the Test set results
ggplot() +
geom_point(aes(x = testset$YearsExperience, y = testset$Salary),
           colour = 'red') +
geom_line(aes(x = trainingset$YearsExperience,
           y = predict(lm.r, newdata = trainingset)),
           colour = 'blue') +
ggtitle('Salary vs Experience (Test set)') +
xlab('Years of experience') +
ylab('Salary')
```

327 Mohit Kapoor

Output:

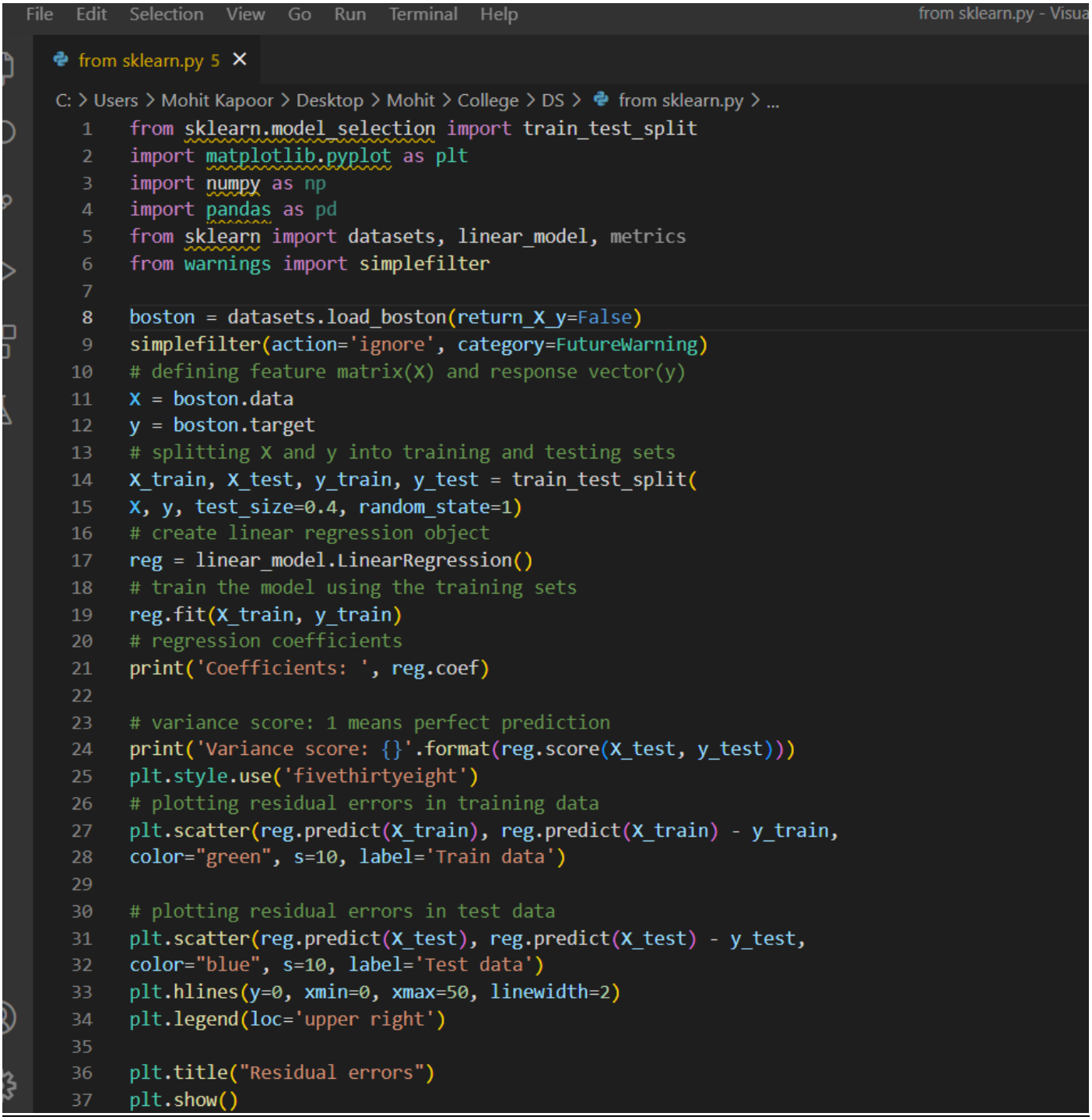
Intercept	YearsExperience
24558.39	10639.23



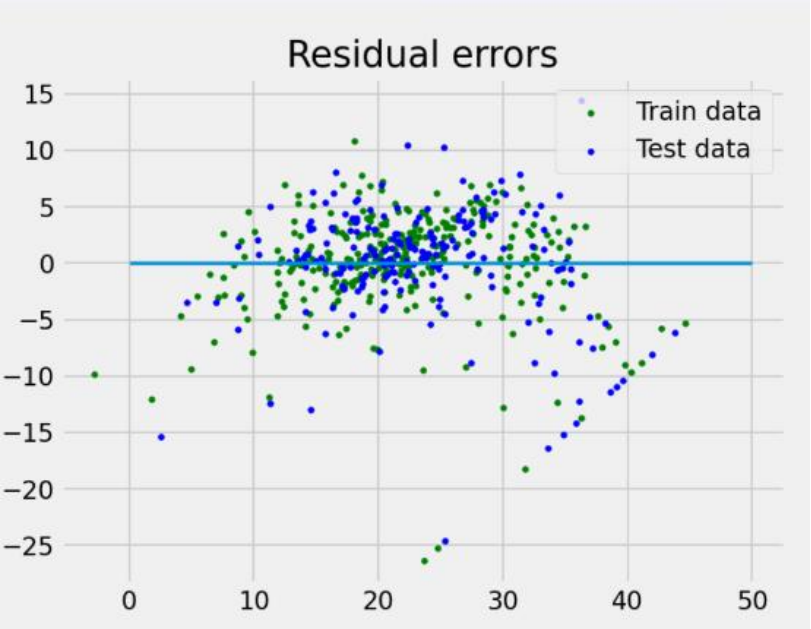


327 Mohit Kapoor

Multiple Linear Regression:



Output:



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

327 Mohit Kapoor

Practical 3

Aim: Demonstration of Logistic Regression

Theory: Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for a given set of features (or inputs), X.

Contrary to popular belief, logistic regression is a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as "1". Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

$$g(z) = \frac{1}{1+e^{-z}}$$

Terminologies involved in Logistic Regression:

Here are some common terminologies involved in logistic regression:

- Dependent variable: The target variable in a logistic regression model, which we are trying to predict.
- Independent variables: The input characteristics or predictor factors applied to the dependent variable's predictions.
- Logistic function: The formula used to represent how the independent and dependent variables relate to one another. The logistic function transforms the input variables into a probability value between 0 and 1, which represents the likelihood of the dependent variable being 1 or 0.
- Odds: The proportion of an event's chances of happening to its chances of not happening. The chances are used in logistic regression to model the connection between the independent and dependent variables.
- Log-odds: The logistic regression model's calculation is made simpler by using the logarithm of the odds.
- Coefficient: The logistic regression model's estimated parameters, which show how the independent and dependent variables relate to one another.
- Intercept: A constant term in the logistic regression model, which represents the log-odds when all independent variables are equal to zero.
- Maximum likelihood estimation: The method used to estimate the coefficients of the logistic regression model, which maximizes the likelihood of observing the data given the model.
- Confusion matrix: A table that lists the number of true positive, true negative, false positive, and false negative predictions made by a logistic regression model is used to assess the model's performance.

Dataset: Here, I have used a weather dataset which is to be used in logistics regression.

H14

	A	B	C	D	E	F	G
ATA	outlook	temp	humidity	windy	play golf		
2	rainy	hot	high	FALSE	no		
3	rainy	hot	high	TRUE	no		
4	overcast	hot	high	FALSE	yes		
5	sunny	mild	high	FALSE	yes		
6	sunny	cool	normal	FALSE	yes		
7	sunny	cool	normal	TRUE	no		
8	overcast	cool	normal	TRUE	yes		
9	rainy	mild	high	FALSE	yes		
10	rainy	cool	normal	FALSE	yes		
11	sunny	mild	normal	FALSE	yes		
12	rainy	mild	normal	TRUE	yes		
13	overcast	mild	high	TRUE	yes		
14	overcast	hot	normal	FALSE	yes		
15	sunny	mild	high	TRUE	no		
16							
17							
18							
19							

327 Mohit Kapoor

Steps:

1. Store the dataset in a variable.

```
> X=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/weather1.csv")
> x
Error: object 'x' not found
> X
    outlook temp humidity windy play.golf
1     rainy  hot     high FALSE        no
2     rainy  hot     high  TRUE        no
3  overcast  hot     high FALSE       yes
4     sunny mild     high FALSE       yes
5     sunny cool   normal FALSE       yes
6     sunny cool   normal  TRUE        no
7  overcast cool   normal  TRUE       yes
8     rainy mild     high FALSE       yes
9     rainy cool   normal FALSE       yes
10    sunny mild   normal FALSE       yes
11    rainy mild   normal  TRUE       yes
12 overcast mild     high  TRUE       yes
13 overcast  hot   normal FALSE       yes
14    sunny mild     high  TRUE        no
> |
```

2. Printing the dataset.

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

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

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

3. Partitioning the dataset.

```
> s=sample(nrow(X),.7*nrow(X))
> x_tr=x[s]
Error: object 'x' not found
> X_tr=X[s,]
> X_test=X[-s,]
> nrow(X)
[1] 14
> nrow(X_tr)
[1] 9
> nrow(X_test)
[1] 5
> 
```

327 Mohit Kapoor

4. Data Modeling

```
> 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
      1.609       -2.303

Degrees of Freedom: 8 Total (i.e. Null);  7 Residual
Null Deviance:      11.46
Residual Deviance: 9.226        AIC: 13.23
>
```

5. Taking the summary

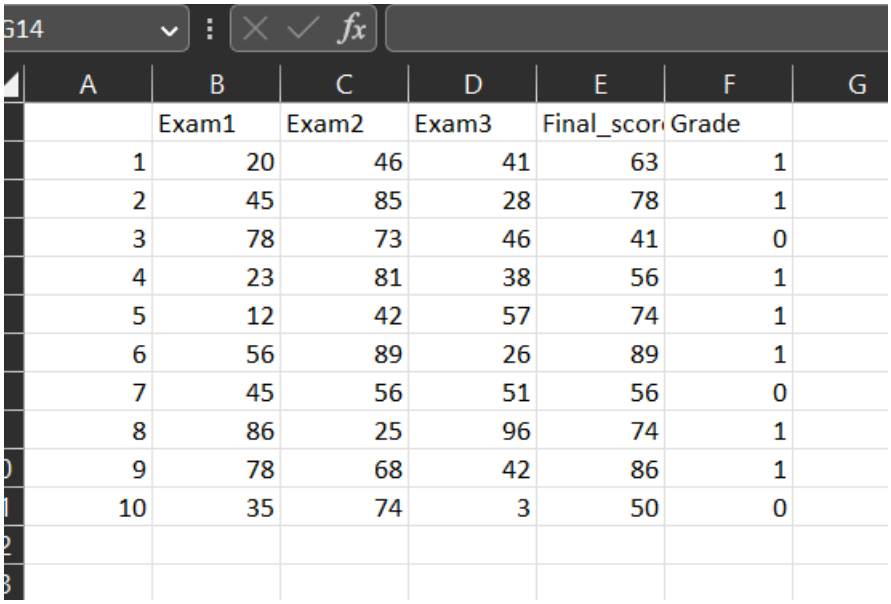
```
> summary(lmod)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8930  -0.9005   0.6039   0.6039   1.4823

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.609      1.095   1.469    0.142
windy         -2.303      1.643  -1.401    0.161

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 11.4573  on 8  degrees of freedom
Residual deviance:  9.2258  on 7  degrees of freedom
AIC: 13.226

Number of Fisher Scoring iterations: 4

>
> 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
-1.17741  -1.17741   0.00008   1.17741   1.17741

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    19.57    6208.83   0.003    0.997
humidity      -19.57    6208.83  -0.003    0.997

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 11.4573  on 8  degrees of freedom
Residual deviance:  8.3178  on 7  degrees of freedom
AIC: 12.318

Number of Fisher Scoring iterations: 18
```

327 Mohit Kapoor

4. Prediction of the dataset

```
> p=predict(lmod,X_test,type="response")
> p
  3   6   7  11  13
0.5 1.0 1.0 1.0 1.0
> q()
```

Second dataset:



	A	B	C	D	E	F	G
		Exam1	Exam2	Exam3	Final_scor	Grade	
	1	20	46	41	63	1	
	2	45	85	28	78	1	
	3	78	73	46	41	0	
	4	23	81	38	56	1	
	5	12	42	57	74	1	
	6	56	89	26	89	1	
	7	45	56	51	56	0	
	8	86	25	96	74	1	
	9	78	68	42	86	1	
	10	35	74	3	50	0	

1. Storing a dataset in a variable

```
> x3=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/grade.csv")
> x3
    X Exam1 Exam2 Exam3 Final_score Grade
1   1    20    46    41          63     1
2   2    45    85    28          78     1
3   3    78    73    46          41     0
4   4    23    81    38          56     1
5   5    12    42    57          74     1
6   6    56    89    26          89     1
7   7    45    56    51          56     0
8   8    86    25    96          74     1
9   9    78    68    42          86     1
10 10    35    74     3          50     0
```

327 Mohit Kapoor

```
> 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
```

2. Predicting the data in 1's and 0's form

```
> 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
```

3. Prediction matrix

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

4. Actuals predicted

```
> 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
```

5. Variable influence factor

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

Conclusion: Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. Therefore, it is very easy to predict values or outcomes of datasets in small amounts that can be easily solved.

Practical 4

Aim: Demonstration of Hypothesis Testing.

Theory: Hypothesis testing in statistics refers to analysing an assumption about a population parameter. It is used to make an educated guess about an assumption using statistics. With the use of sample data, hypothesis testing makes an assumption about how true the assumption is for the entire population from where the sample is being taken. Any hypothetical statement we make may or may not be valid, and it is then our responsibility to provide evidence for its possibility. To approach any hypothesis, we follow these four simple steps that test its validity. First, we formulate two hypothetical statements such that only one of them is true. By doing so, we can check the validity of our own hypothesis. The next step is to formulate the statistical analysis to be followed based upon the data points. Then we analyse the given data using our methodology. The final step is to analyse the result and judge whether the null hypothesis will be rejected or is true.

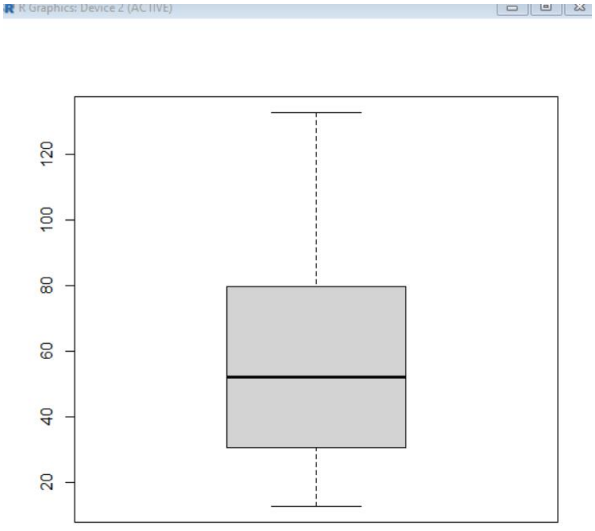
Step 1: First we have to create Excel file and enter the 28 values so that we can find deviation, Square of deviation, population, differentiate of mean, T-value, and system calculate standard deviation and save as .CSV file.

	A	B	C	D	E
1	C1				
2	75.3				
3	36.1				
4	59.5				
5	19.4				
6	98.6				
7	14.8				
8	45.9				
9	87.6				
10	54.9				
11	28.9				
12	42.7				
13	33.8				
14	25.7				
15	12.8				
16	95.4				
17	23.4				
18	52				
19	95.1				
20	62.3				
21	74.8				
22	85.2				
23	52.4				
24	32.2				
25	17.5				
26	65.1				
27	35.8				
28	132.6				
29	84.1				

Step 2: Now we have to import Excel file (327.csv)

```
> datanew=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/327.csv")
> datanew
      C1
1   75.3
2   36.1
3   59.5
4   19.4
5   98.6
6   14.8
7   45.9
8   87.6
9   54.9
10  28.9
11  42.7
12  33.8
13  25.7
14  12.8
15  95.4
16  23.4
17  52.0
18  95.1
19  62.3
20  74.8
21  85.2
22  52.4
23  32.2
24  17.5
25  65.1
26  35.8
27 132.6
28  84.1
```

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



Step 4: After that find the mean of respective data.

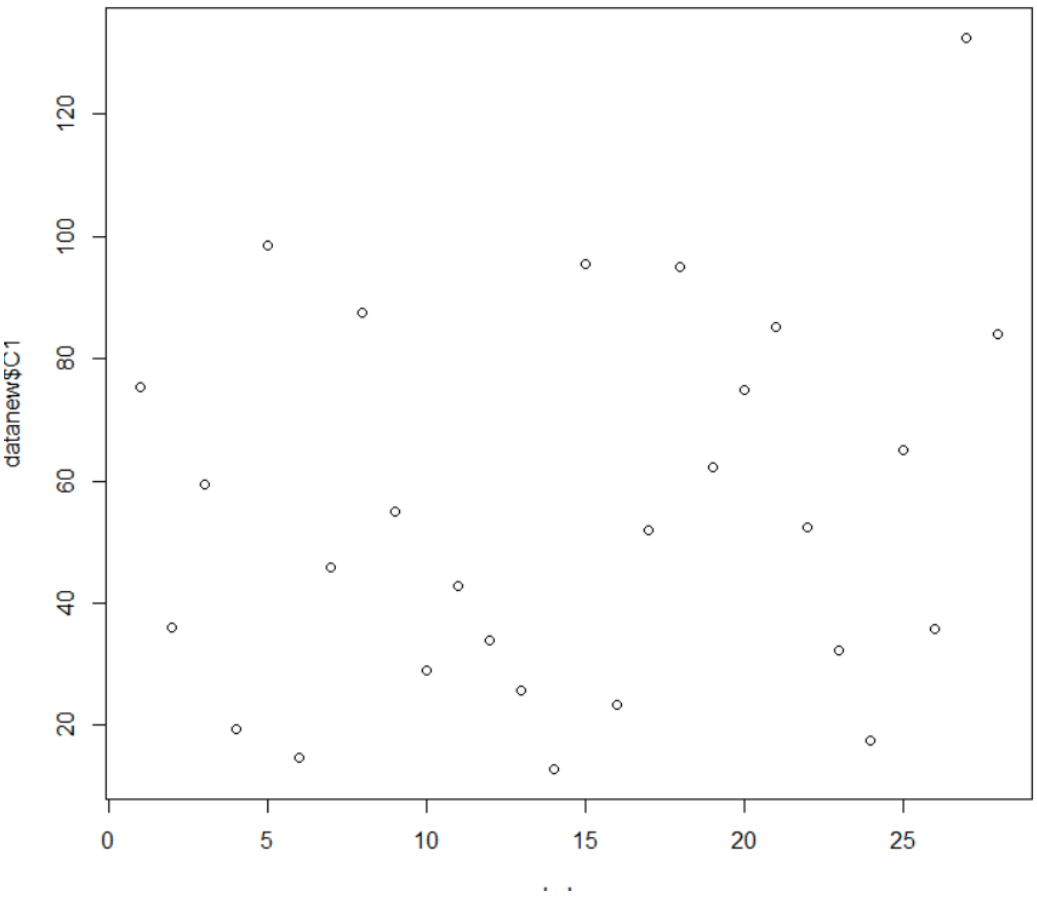
```
> boxplot(datanew)
> mean1=mean(datanew$C1)
> mean1
[1] 55.13929
```

Step 5: Now we have to calculate the standard deviation.

```
> standev1=sd(datanew$C1)
> standev1
[1] 30.74229
```

Step 6: Plotting the bell curve.

```
> plot(datanew$C1)
> q()
```



Step 7: Finally, at the end find the T-test value.

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

        One Sample t-test

data:  datanew$C1
t = -7.7216, df = 27, p-value = 1
alternative hypothesis: true mean is greater than 100
95 percent confidence interval:
 45.24361      Inf
sample estimates:
mean of x
 55.13929
```

Conclusion: Thus we have concluded the hypothesis testing of the above dataset, having the following values:

Mean: 55.13929

Standard deviation: 30.74229

T-test: t = -7.7216, df = 27, p=value = 1.

327 Mohit Kapoor

Practical 5

Aim: Demonstration of Analysis of Variance

Theory: Analysis of variance (ANOVA) is an analysis tool used in statistics that splits an observed aggregate variability found inside a data set into two parts: systematic factors and random factors. The systematic factors have a statistical influence on the given data set, while the random factors do not. Analysts use the ANOVA test to determine the influence that independent variables have on the dependent variable in a regression study.

Analysis of variance, or ANOVA, is a statistical method that separates observed variance data into different components to use for additional tests.

A one-way ANOVA is used for three or more groups of data, to gain information about the relationship between the dependent and independent variables.

If no true variance exists between the groups, the ANOVA's F-ratio should equal close to 1.

The Formula for Analysis of Variance is:

F= MST/MSE

Where:

F=ANOVA coefficient

MST= Mean sum of squares due to treatment

MSE= Mean sum of squares due to error

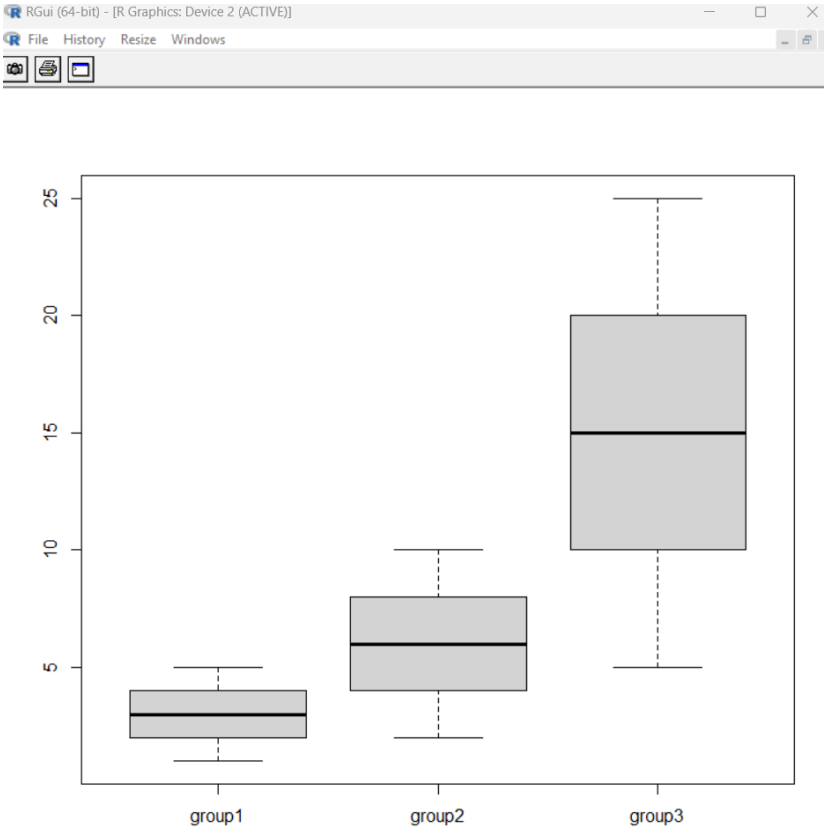
Code:

Dividing the data into 3 groups and printing the boxplot:

```
> group1=c(1,2,3,4,5)
> group2=c(2,4,6,8,10)
> group3=c(5,10,15,20,25)
> cg=data.frame(cbind(group1,group2,group3))
> cg
  group1 group2 group3
1      1      2      5
2      2      4     10
3      3      6     15
4      4      8     20
5      5     10     25
> boxplot(cg)
```

327 Mohit Kapoor

Boxplot:



Printing the data into stack format:

```
> stacked_g=stack(cg)
> stacked_g
   values    ind
1       1 group1
2       2 group1
3       3 group1
4       4 group1
5       5 group1
6       2 group2
7       4 group2
8       6 group2
9       8 group2
10     10 group2
11      5 group3
12     10 group3
13     15 group3
14     20 group3
15     25 group3
```

327 Mohit Kapoor
Taking another dataset and repeating all the steps:

R Console

```
> av=aov(values~ind,data=stacked_g)
> summary(av)
            Df Sum Sq Mean Sq F value  Pr(>F)
ind          2    390     195     7.8 0.00676 **
Residuals   12    300      25
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> g1=c(50,51,52,53,54)
> g2=c(45,46,47,48,49)
> g3=c(31,32,33,34,35)
> cg1=data.frame(cbind(g1,g2,g3))
> cg1
  g1 g2 g3
1 50 45 31
2 51 46 32
3 52 47 33
4 53 48 34
5 54 49 35
> stacked_g=stack(cg1)
> stacked_g
   values ind
1      50  g1
2      51  g1
3      52  g1
4      53  g1
5      54  g1
6      45  g2
7      46  g2
8      47  g2
9      48  g2
10     49  g2
11     31  g3
12     32  g3
13     33  g3
14     34  g3
15     35  g3
```

```
> av=aov(values~ind,data=stacked_g)
> av1=aov(values~ind, data=stacked_g)
> summary(av1)
            Df Sum Sq Mean Sq F value   Pr(>F)
ind          2    970   485.0     194 7.29e-10 ***
Residuals   12     30     2.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

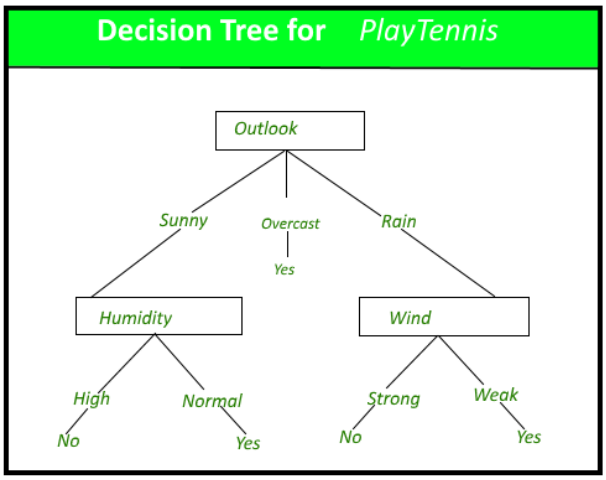
Conclusion: Analysis of variance (ANOVA) is the most powerful analytic tool available in statistics. It splits an observed aggregate variability that is found inside the data set. Then separate the data into systematic factors and random factors. In the systematic factor, that data set has statistical influence.

327 Mohit Kapoor

Practical 6

Aim: Demonstration of Decision Tree

Theory: Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Strengths and Weaknesses of the Decision Tree approach

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.
- Ease of use: Decision trees are simple to use and don't require a lot of technical expertise, making them accessible to a wide range of users.
- Scalability: Decision trees can handle large datasets and can be easily parallelized to improve processing time.
- Missing value tolerance: Decision trees are able to handle missing values in the data, making them a suitable choice for datasets with missing or incomplete data.
- Handling non-linear relationships: Decision trees can handle non-linear relationships between variables, making them a suitable choice for complex datasets.
- Ability to handle imbalanced data: Decision trees can handle imbalanced datasets, where one class is heavily represented compared to the others, by weighting the importance of individual nodes based on the class distribution.

327 Mohit Kapoor

Code:

Steps:

Step1: click on packages and set cran mirror.

Step2: click on packages and select install packages and install 3 packages (rpart,tree,rattle)

Step3:(OPTIONAL Application for version 4.2)

install.packages("rpart")

install.packages("tree")

install.packages("rattle")

Step4: Create a dataset with .csv extension.

Step5: Read excel data in R GUI.

```
> x=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/newweather.csv")
> x
    outlook temp humidity windy play.golf
1     sunny  hot     high FALSE       yes
2     sunny  hot   normal  TRUE       yes
3  overcast  hot   normal FALSE       yes
4     sunny mild     high  TRUE        no
5     sunny mild   normal FALSE       yes
6     sunny mild     high  TRUE        no
7     rainy cool   normal  TRUE        no
8     rainy mild     high FALSE        no
9  overcast cool   normal FALSE        no
10    sunny mild   normal  TRUE       yes
11    rainy mild     high  TRUE        no
12 overcast mild     high  TRUE       yes
13    rainy  hot     high FALSE       yes
14    sunny mild   normal  TRUE        no
```

Step6: Create Sample partition of excel data.

Create a weather partition for training.

Create a weather partition for testing.

```
> sample_weather=sample(nrow(x),.7*nrow(x))
> weather_tr=x[sample_weather,]
> weather_test=x[-sample_weather,]
> weather_test
    outlook temp humidity windy play.golf
3  overcast  hot   normal FALSE       yes
4     sunny mild     high  TRUE        no
8     rainy mild     high FALSE        no
10    sunny mild   normal  TRUE       yes
14    sunny mild   normal  TRUE        no
```

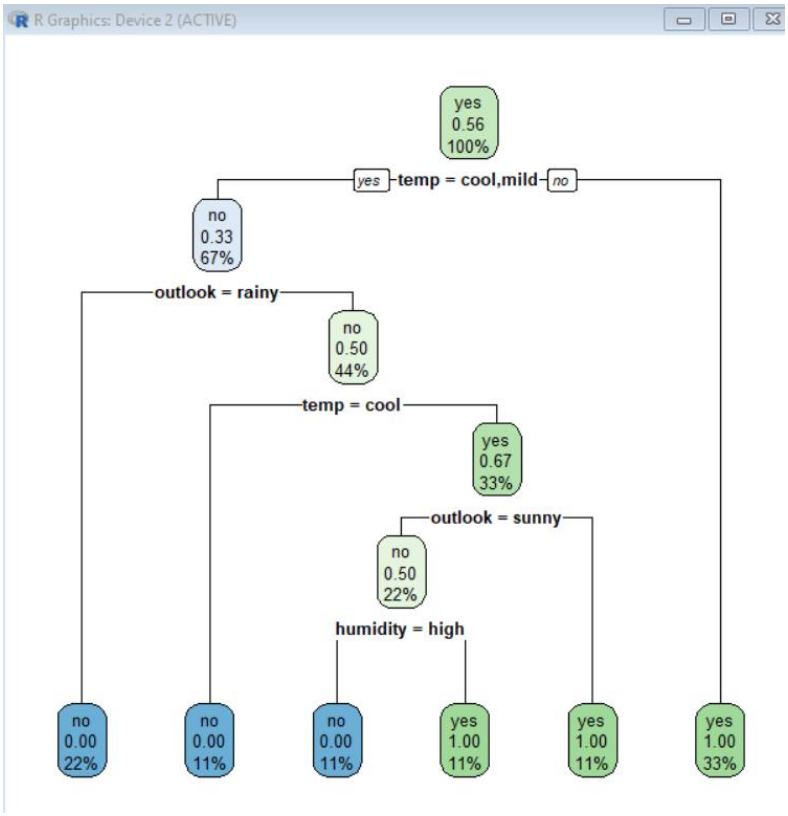
Step7: Call rpart packages

```
> library(rpart)
> library(rpart.plot)
```

Step 8: Plot the decision tree

327 Mohit Kapoor

```
> dtreemod2=rpart(Hours.Played~.,data=weather_tr2,method="anova",control=rpart.control(minsplit=1,minbucket=1))
```



Step9: Predict Tree

```
> rpart.plot(dtreemod)
> p=predict(dtreemod,weather_test,type="class")
> weather_test
    outlook temp humidity windy play.golf
3  overcast  hot   normal FALSE       yes
4     sunny mild     high  TRUE        no
8     rainy mild     high FALSE        no
10    sunny mild   normal  TRUE       yes
14    sunny mild   normal  TRUE        no
> table(weather_test$play.gold,p)
Error in table(weather_test$play.gold, p) :
  all arguments must have the same length
> table(weather_test$play.golf,p)
     p
      no yes
  no   2   1
  yes  0   2
```

Step10: Printing Rules with rpart.rules

327 Mohit Kapoor

```
> rpart.rules(dtreemod)
 play.golf
      0.00 when temp is cool or mild & outlook is             rainy
      0.00 when temp is         cool & outlook is overcast or sunny
      0.00 when temp is         mild & outlook is             sunny & humidity is   high
      1.00 when temp is         mild & outlook is             sunny & humidity is normal
      1.00 when temp is         mild & outlook is          overcast
      1.00 when temp is          hot
```

Regression tree:

Step11: Read the data into a variable

```
> x2=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/weather2.csv")
> x2
    Outlook  temp Humidity Windy Hours.Played
1     Rainy  Hot      High FALSE           26
2     Rainy  Hot      High  TRUE           30
3  Overcast  Hot      High FALSE           48
4     Sunny  Mild     High FALSE           46
5     Sunny  Cool   Normal FALSE           62
6  Overcast  Cool   Normal  TRUE           43
7     Rainy  Mild     High FALSE           36
8     Rainy  Cool   Normal FALSE           38
9     Sunny  Mild   Normal FALSE           48
10    Rainy  Mild   Normal  TRUE           48
11 Overcast  Mild     High  TRUE           62
12 Overcast   Hot   Normal FALSE           44
13    Sunny  Mild     High  TRUE           30
```

Step12: Do all the above steps and plot the decision tree

```
> weather_tr2=x2[S2,]
Error in `[.data.frame`(x2, S2, ) : object 'S2' not found
> s2=sample(nrow(x),.7*nrow(x))
> weather_tr2=x2[s2,]
> weather_test2=x2[-s2,]
> weather_test2
    Outlook temp Humidity Windy Hours.Played
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
> libary(rpart)
Error in libary(rpart) : could not find function "libary"
> library(rpart)
> library(rpart.plot)
> dtreemod2=rpart(Hours.Played~.,data=weather_tr2,method="anova",control=rpart.control(minsplit=1,minbucket=1))
> rpart.plot(dtreemod2)
> rpart.rules(dtreemod2)
 Hours.Played
          28 when Outlook is             Rainy
          30 when Outlook is             Sunny & Windy is 1
          43 when Outlook is          Overcast & Windy is 1
          46 when Outlook is Overcast or Sunny & Windy is 0 & temp is  Hot  or Hot or Mild
          62 when Outlook is Overcast or Sunny & Windy is 0 & temp is                 Cool
```

327 Mohit Kapoor

Step13: Prediction

```
> actuals_preds<-data.frame(cbind(actuals=weather_test2@Hours.played,predicts=p))
Error in cbind(actuals = weather_test2@Hours.played, predicts = p) :
  trying to get slot "Hours.played" from an object (class "data.frame") that is not an S4 object
> actuals_preds<-data.frame(cbind(actuals=weather_test2$Hours.played,predicts=p))
> actuals_preds
   predicts
3         2
4         1
8         1
10        2
14        2
>
```

Conclusion: Decision trees are one of the best forms of learning algorithms based on various learning methods. They boost predictive models with accuracy, ease in interpretation, and stability. The tools are also effective in fitting non-linear relationships since they can solve data-fitting challenges, such as regression and classifications.

327 Mohit Kapoor

Practical 7

Aim: Demonstration of Principal Component Analysis

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

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

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

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

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

Dimensionality: It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.

Correlation: It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

Orthogonal: It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

Eigenvectors: If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.

Covariance Matrix: A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Code:

Step1: click on packages and set cran mirror(click on other and select USA IN)

Step2: click on packages and select install packages and install package FactoMineR.

```
install.packages("FactoMineR")
```

```
library(FactoMineR)
```

Step3: Create a database in excel

Step4: Read the database into a variable

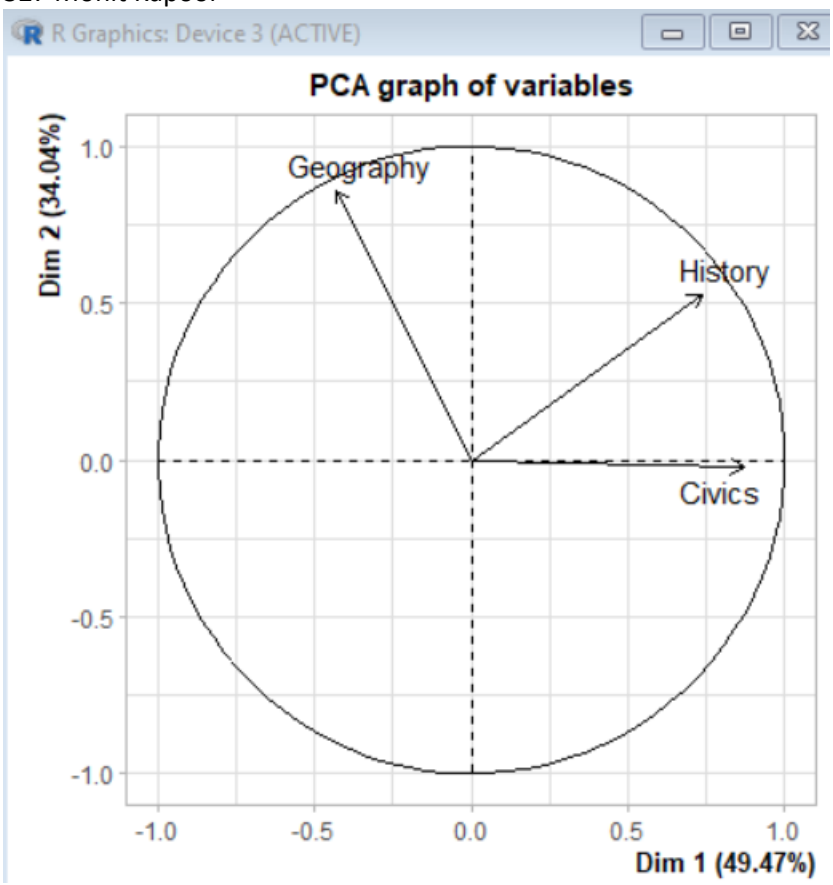
327 Mohit Kapoor

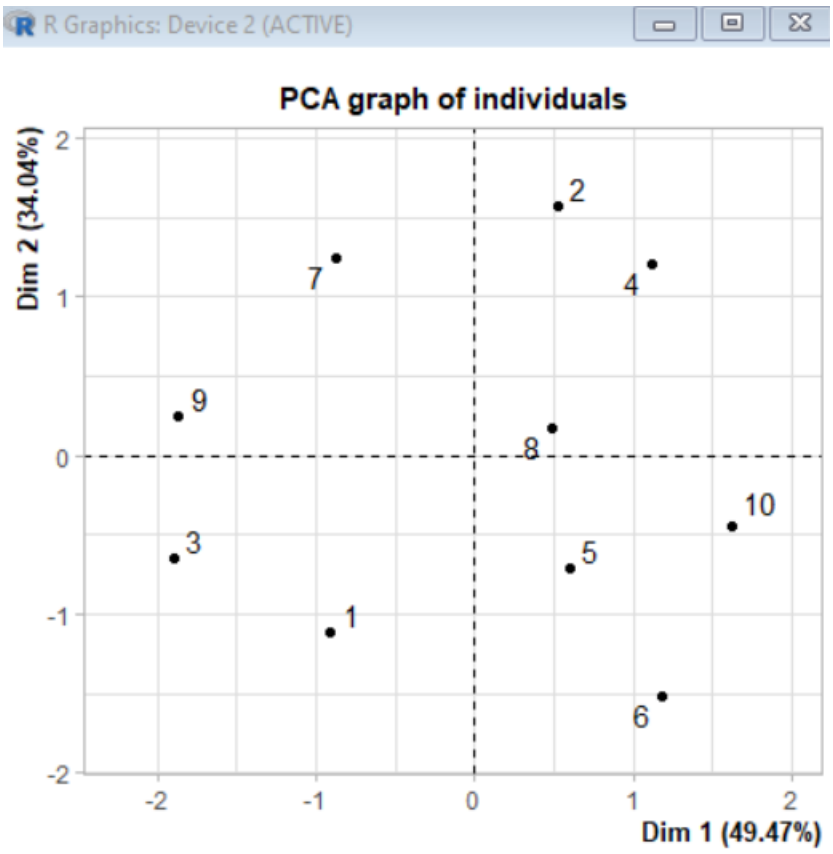
```
> x=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/datarecord.csv")
> x
   History Civics Geography
1       45     24        43
2       68     85        96
3       14     42        75
4       86     74        75
5       57     69        46
6       53     85        29
7       69     25        86
8       78     46        54
9       43     12        78
10      74     86        42
```

Step5: Plot the PCA graph.

```
> cov_mat=cov(x)
> cov_mat
             History    Civics  Geography
History   445.78889  252.8222   11.24444
Civics    252.82222  808.6222 -164.68889
Geography  11.24444 -164.6889  499.37778
> ex=eigen(cov_mat)
> ex
eigen() decomposition
$values
[1] 980.8617 494.8927 278.0345

$vectors
           [,1]       [,2]       [,3]
[1,] -0.4042222 0.49397362  0.7698016
[2,] -0.8682847 0.05735449 -0.4927394
[3,]  0.2875518 0.86758315 -0.4057258

> datapca=PCA(x,ncp=3,graph=TRUE)
```

327 Mohit Kapoor





327 Mohit Kapoor
Step6: Compute eig,var and coord

```
> datapca$eig
       eigenvalue percentage of variance cumulative percentage of variance
comp 1  1.4840534                49.46845                           49.46845
comp 2  1.0212659                34.04220                           83.51064
comp 3  0.4946807                16.48936                          100.00000
> datapca$var
$coord
               Dim.1       Dim.2      Dim.3
History    0.7360300  0.52875459 -0.4227037
Civics     0.8704028 -0.02202159  0.4918476
Geography -0.4297815  0.86092947  0.2721916

$cor
               Dim.1       Dim.2      Dim.3
History    0.7360300  0.52875459 -0.4227037
Civics     0.8704028 -0.02202159  0.4918476
Geography -0.4297815  0.86092947  0.2721916

$cos2
              Dim.1        Dim.2      Dim.3
History   0.5417402 0.2795814188 0.17867839
Civics    0.7576010 0.0004849506 0.24191406
Geography 0.1847122 0.7411995569 0.07408827

$contrib
             Dim.1       Dim.2    Dim.3
History   36.50409 27.37596659 36.11994
Civics    51.04944  0.04748524 48.90307
Geography 12.44646 72.57654817 14.97699

> datapca$var$coord
               Dim.1       Dim.2      Dim.3
History    0.7360300  0.52875459 -0.4227037
Civics     0.8704028 -0.02202159  0.4918476
Geography -0.4297815  0.86092947  0.2721916
```

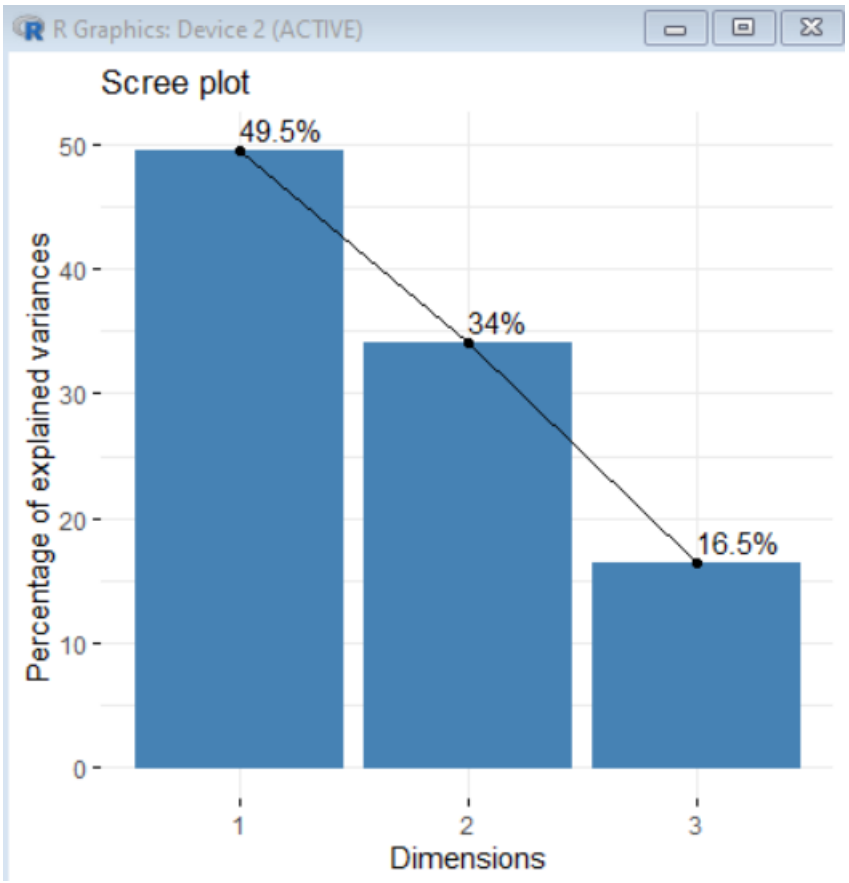
Step7: Install the factoextra package

```
> install.packages('factoextra',repos="http://cran.us.r-project.org")
Installing package into 'C:/Users/Mohit Kapoor/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
also installing the dependencies 'corrplot', 'viridis', 'ggsci', 'cowplot', 'gg$

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/4.2/corrplot_0.92.$
Content type 'application/zip' length 3844837 bytes (3.7 MB)
downloaded 3.7 MB

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/4.2/viridis_0.6.2.$
Content type 'application/zip' length 2999943 bytes (2.9 MB)
downloaded 2.9 MB

trying URL 'http://cran.us.r-project.org/bin/windows/contrib/4.2/ggsci_2.9.zip'
Content type 'application/zip' length 2978365 bytes (2.8 MB)
downloaded 2.8 MB
```

```
package 'dendextend' successfully unpacked and MD5 sums checked
package 'ggpubr' successfully unpacked and MD5 sums checked
package 'reshape2' successfully unpacked and MD5 sums checked
package 'factoextra' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Mohit Kapoor\AppData\Local\Temp\RtmpYnBIOT\downloaded_packages
```

Step8: Plot the fviz

```
> library("factoextra")
Loading required package: ggplot2
Welcome! Want to learn more? See two factoextra-related books at https://goo
> fviz_screeplot(datapca,addlabels=TRUE,ylim=c(0,50))
```



327 Mohit Kapoor
Step9: Plot the PCA graph of iris variables

```
> 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
26           5.0         3.0          1.6         0.2
27           5.0         3.4          1.6         0.4
28           5.2         3.5          1.5         0.2
29           5.2         3.4          1.4         0.2
30           4.7         3.2          1.6         0.2
31           4.8         3.1          1.6         0.2
```

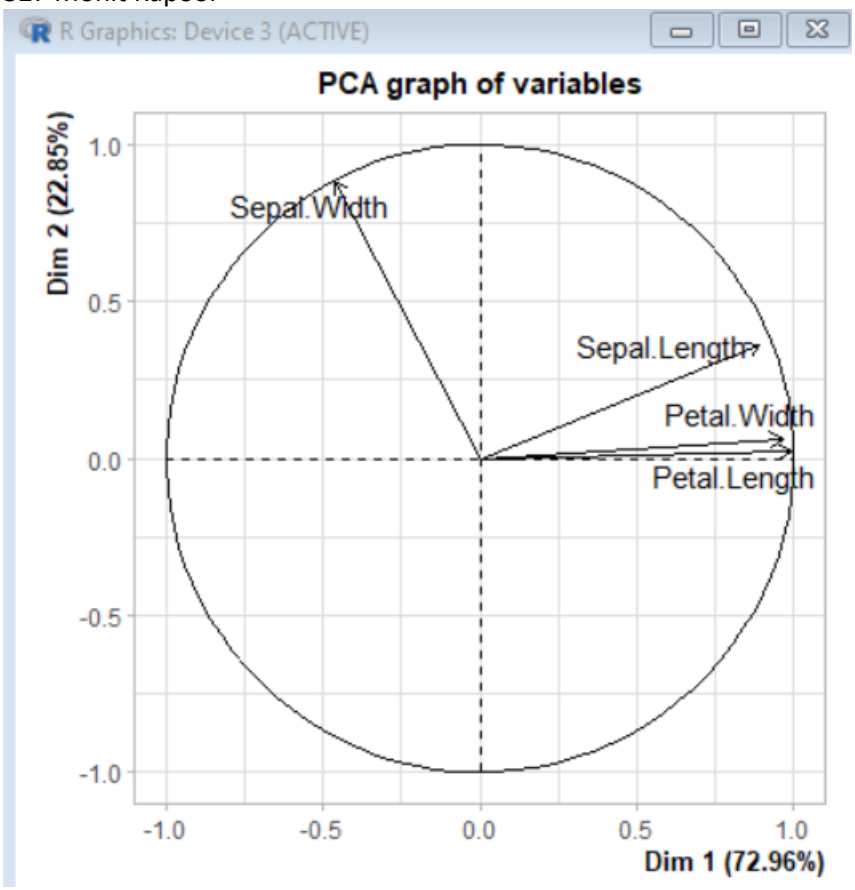
31	4.8	3.1	1.6	0.2
32	5.4	3.4	1.5	0.4
33	5.2	4.1	1.5	0.1
34	5.5	4.2	1.4	0.2
35	4.9	3.1	1.5	0.2
36	5.0	3.2	1.2	0.2
37	5.5	3.5	1.3	0.2
38	4.9	3.6	1.4	0.1
39	4.4	3.0	1.3	0.2
40	5.1	3.4	1.5	0.2
41	5.0	3.5	1.3	0.3
42	4.5	2.3	1.3	0.3
43	4.4	3.2	1.3	0.2
44	5.0	3.5	1.6	0.6
45	5.1	3.8	1.9	0.4
46	4.8	3.0	1.4	0.3
47	5.1	3.8	1.6	0.2
48	4.6	3.2	1.4	0.2
49	5.3	3.7	1.5	0.2
50	5.0	3.3	1.4	0.2
51	7.0	3.2	4.7	1.4
52	6.4	3.2	4.5	1.5
53	6.9	3.1	4.9	1.5
54	5.5	2.3	4.0	1.3
55	6.5	2.8	4.6	1.5
56	5.7	2.8	4.5	1.3
57	6.3	3.3	4.7	1.6
58	4.9	2.4	3.3	1.0
59	6.6	2.9	4.6	1.3
60	5.2	2.7	3.9	1.4
61	5.0	2.0	3.5	1.0
62	5.9	3.0	4.2	1.5
63	6.0	2.2	4.0	1.0
64	6.1	2.9	4.7	1.4
65	5.6	2.9	3.6	1.3
66	6.7	3.1	4.4	1.4
67	5.6	3.0	4.5	1.5
68	5.8	2.7	4.1	1.0
69	6.2	2.2	4.5	1.5
70	5.6	2.5	3.9	1.1
71	5.9	3.2	4.8	1.8
72	6.1	2.8	4.0	1.3

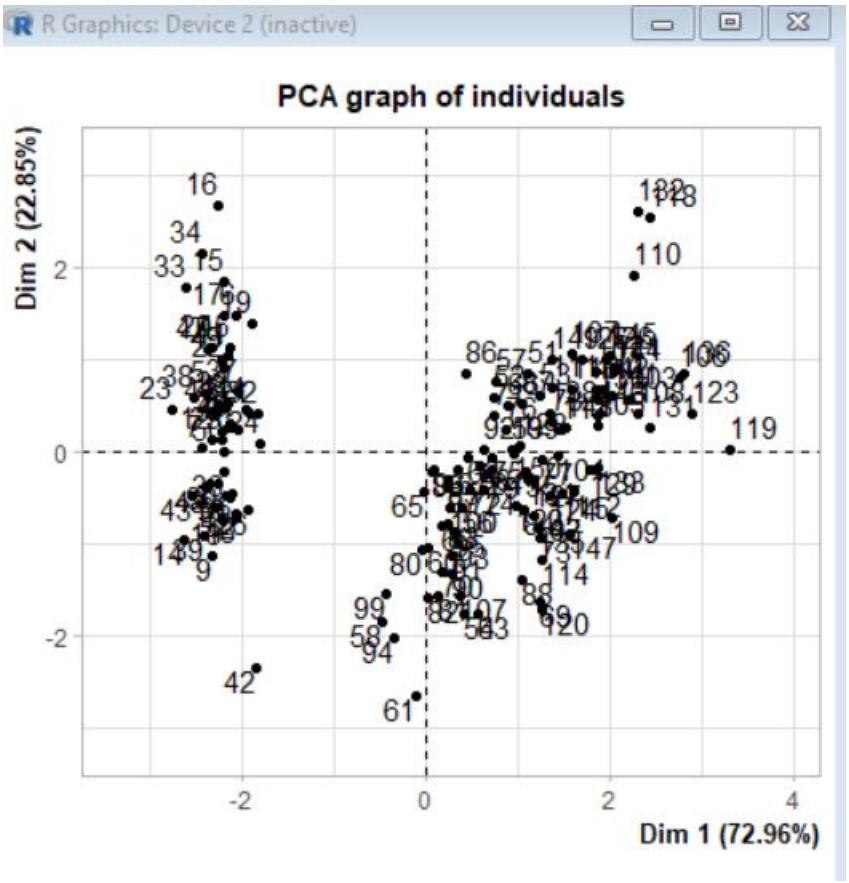
73	6.3	2.5	4.9	1.5
74	6.1	2.8	4.7	1.2
75	6.4	2.9	4.3	1.3
76	6.6	3.0	4.4	1.4
77	6.8	2.8	4.8	1.4
78	6.7	3.0	5.0	1.7
79	6.0	2.9	4.5	1.5
80	5.7	2.6	3.5	1.0
81	5.5	2.4	3.8	1.1
82	5.5	2.4	3.7	1.0
83	5.8	2.7	3.9	1.2
84	6.0	2.7	5.1	1.6
85	5.4	3.0	4.5	1.5
86	6.0	3.4	4.5	1.6
87	6.7	3.1	4.7	1.5
88	6.3	2.3	4.4	1.3
89	5.6	3.0	4.1	1.3
90	5.5	2.5	4.0	1.3
91	5.5	2.6	4.4	1.2
92	6.1	3.0	4.6	1.4
93	5.8	2.6	4.0	1.2
94	5.0	2.3	3.3	1.0
95	5.6	2.7	4.2	1.3
96	5.7	3.0	4.2	1.2
97	5.7	2.9	4.2	1.3
98	6.2	2.9	4.3	1.3
99	5.1	2.5	3.0	1.1
100	5.7	2.8	4.1	1.3
101	6.3	3.3	6.0	2.5
102	5.8	2.7	5.1	1.9
103	7.1	3.0	5.9	2.1
104	6.3	2.9	5.6	1.8
105	6.5	3.0	5.8	2.2
106	7.6	3.0	6.6	2.1
107	4.9	2.5	4.5	1.7
108	7.3	2.9	6.3	1.8
109	6.7	2.5	5.8	1.8
110	7.2	3.6	6.1	2.5
111	6.5	3.2	5.1	2.0
112	6.4	2.7	5.3	1.9
113	6.8	3.0	5.5	2.1

111	6.5	3.2	5.1	2.0
112	6.4	2.7	5.3	1.9
113	6.8	3.0	5.5	2.1
114	5.7	2.5	5.0	2.0
115	5.8	2.8	5.1	2.4
116	6.4	3.2	5.3	2.3
117	6.5	3.0	5.5	1.8
118	7.7	3.8	6.7	2.2
119	7.7	2.6	6.9	2.3
120	6.0	2.2	5.0	1.5
121	6.9	3.2	5.7	2.3
122	5.6	2.8	4.9	2.0
123	7.7	2.8	6.7	2.0
124	6.3	2.7	4.9	1.8
125	6.7	3.3	5.7	2.1
126	7.2	3.2	6.0	1.8
127	6.2	2.8	4.8	1.8
128	6.1	3.0	4.9	1.8
129	6.4	2.8	5.6	2.1
130	7.2	3.0	5.8	1.6
131	7.4	2.8	6.1	1.9
132	7.9	3.8	6.4	2.0
133	6.4	2.8	5.6	2.2
134	6.3	2.8	5.1	1.5
135	6.1	2.6	5.6	1.4
136	7.7	3.0	6.1	2.3
137	6.3	3.4	5.6	2.4
138	6.4	3.1	5.5	1.8
139	6.0	3.0	4.8	1.8
140	6.9	3.1	5.4	2.1
141	6.7	3.1	5.6	2.4
142	6.9	3.1	5.1	2.3
143	5.8	2.7	5.1	1.9
144	6.8	3.2	5.9	2.3
145	6.7	3.3	5.7	2.5
146	6.7	3.0	5.2	2.3
147	6.3	2.5	5.0	1.9
148	6.5	3.0	5.2	2.0
149	6.2	3.4	5.4	2.3
150	5.9	3.0	5.1	1.8

```
> cov_iris=cov(x)
> cov_iris
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
> irispca=PCA(x,ncp=3,graph=TRUE)
```

327 Mohit Kapoor





327 Mohit Kapoor
Step10: Show the summary of irispca.

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

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

> summary(irispca)

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


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

Individuals (the 10 first)
                 Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    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 |
```

327 Mohit Kapoor

```
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: The principal component analysis is a widely used unsupervised learning method to perform dimensionality reduction.

Principal Component Analysis (PCA) is a popular statistical technique used to reduce the dimensions of a large data set. It is commonly used in data exploration and pre-processing before model building. In other words, PCA can turn a group of correlated variables into a smaller set of uncorrelated variables known as principal components.

327 Mohit Kapoor

Practical 8

Aim: Demonstration of Clustering

Theory: Clustering is a Machine Learning technique that involves the grouping of data points. Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties and/or features, while data points in different groups should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields.

In Data Science, we can use clustering analysis to gain some valuable insights from our data by seeing what groups the data points fall into when we apply a clustering algorithm.

K-Means Clustering

K-Means is probably the most well-known clustering algorithm. It's taught in a lot of introductory data science and machine learning classes.

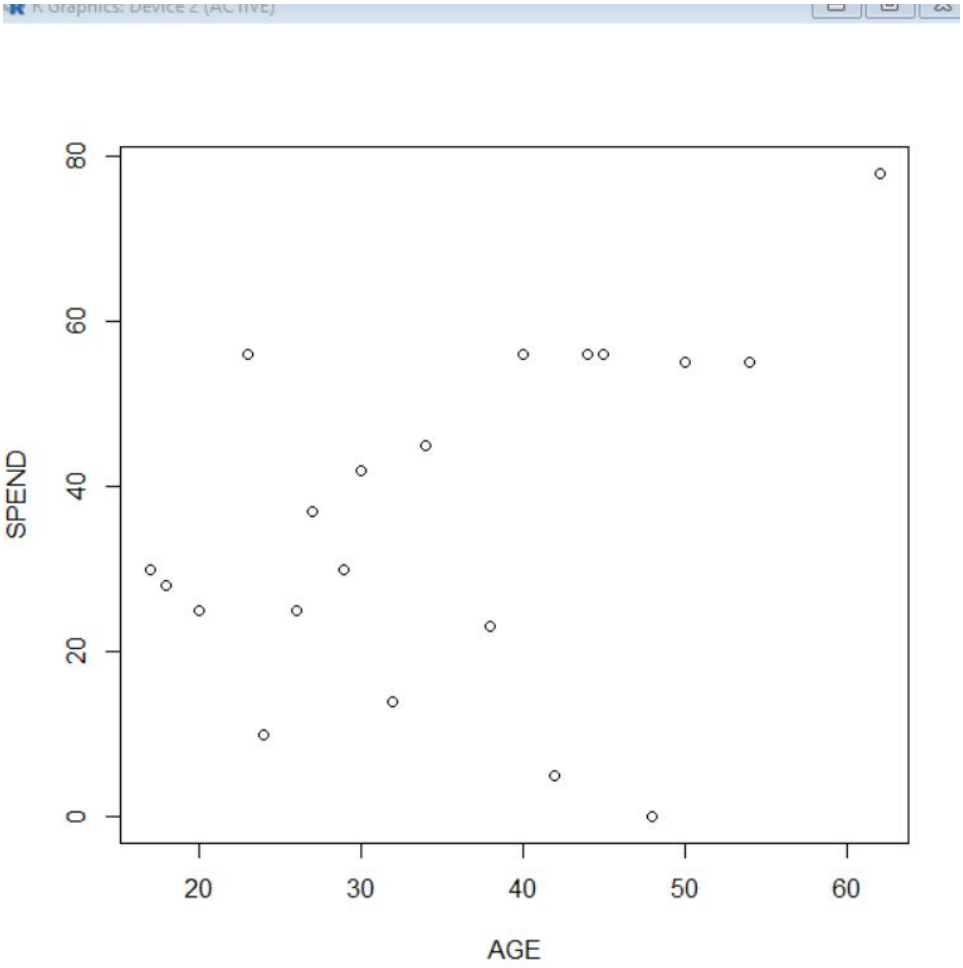
1. To begin, we first select a number of classes/groups to use and randomly initialize their respective center points. To figure out the number of classes to use, it's good to take a quick look at the data and try to identify any distinct groupings. The center points are vectors of the same length as each data point vector and are the "X's" in the graphic above.
2. Each data point is classified by computing the distance between that point and each group center, and then classifying the point to be in the group whose center is closest to it.
3. Based on these classified points, we recompute the group center by taking the mean of all the vectors in the group.
4. Repeat these steps for a set number of iterations or until the group centers don't change much between iterations. You can also opt to randomly initialize the group centers a few times, and then select the run that looks like it provided the best results.

Code:

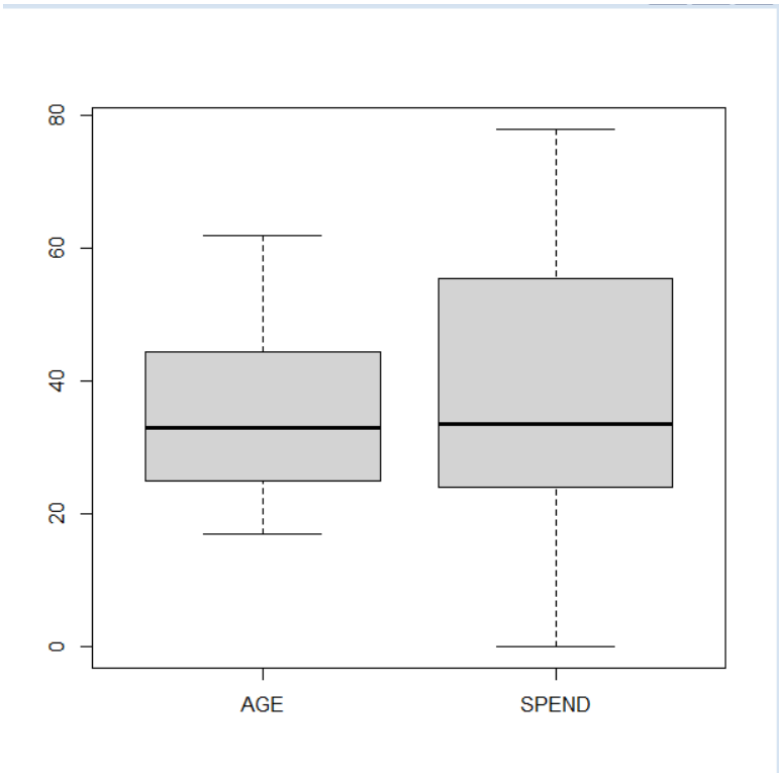
Read the dataset into a data frame.

```
> df=read.csv("C:/Users/Mohit Kapoor/Desktop/Mohit/College/DS/AGE.csv")
> df
   AGE SPEND
1   18    28
2   20    25
3   29    30
4   24    10
5   26    25
6   17    30
7   30    42
8   32    14
9   34    45
10  62    78
11  38    23
12  40    56
13  42     5
14  44    56
15  23    56
16  48     0
17  50    55
18  27    37
19  54    55
20  45    56
```

327 Mohit Kapoor
Plot the data frame



Boxplot the dataframe



327 Mohit Kapoor
Make the Cluster

```
> set.seed(20)
> cl=kmeans(df[,1:2],3)
> cl
K-means clustering with 3 clusters of sizes 9, 6, 5

Cluster means:
       AGE    SPEND
1 24.88889 35.33333
2 49.16667 59.33333
3 36.80000 10.40000

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

Within cluster sum of squares by cluster:
[1] 1160.8889  736.1667  650.0000
 (between_SS / total_SS =  77.3 %)

Available components:

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

Show the dataset

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
16           5.7         4.4          1.5         0.4     setosa
17           5.4         3.9          1.3         0.4     setosa
18           5.1         3.5          1.4         0.3     setosa
19           5.7         3.8          1.7         0.3     setosa
20           5.1         3.8          1.5         0.3     setosa
21           5.4         3.4          1.7         0.2     setosa
22           5.1         3.7          1.5         0.4     setosa
23           4.6         3.6          1.0         0.2     setosa
24           5.1         3.3          1.7         0.5     setosa
25           4.8         3.4          1.9         0.2     setosa
26           5.0         3.0          1.6         0.2     setosa
27           5.0         3.4          1.6         0.4     setosa
28           5.2         3.5          1.5         0.2     setosa
29           5.2         3.4          1.4         0.2     setosa
30           4.7         3.2          1.6         0.2     setosa
31           4.8         3.1          1.6         0.2     setosa
32           5.4         3.4          1.5         0.4     setosa
```

33	5.2	4.1	1.5	0.1	setosa
34	5.5	4.2	1.4	0.2	setosa
35	4.9	3.1	1.5	0.2	setosa
36	5.0	3.2	1.2	0.2	setosa
37	5.5	3.5	1.3	0.2	setosa
38	4.9	3.6	1.4	0.1	setosa
39	4.4	3.0	1.3	0.2	setosa
40	5.1	3.4	1.5	0.2	setosa
41	5.0	3.5	1.3	0.3	setosa
42	4.5	2.3	1.3	0.3	setosa
43	4.4	3.2	1.3	0.2	setosa
44	5.0	3.5	1.6	0.6	setosa
45	5.1	3.8	1.9	0.4	setosa
46	4.8	3.0	1.4	0.3	setosa
47	5.1	3.8	1.6	0.2	setosa
48	4.6	3.2	1.4	0.2	setosa
49	5.3	3.7	1.5	0.2	setosa
50	5.0	3.3	1.4	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
56	5.7	2.8	4.5	1.3	versicolor
57	6.3	3.3	4.7	1.6	versicolor
58	4.9	2.4	3.3	1.0	versicolor
59	6.6	2.9	4.6	1.3	versicolor
60	5.2	2.7	3.9	1.4	versicolor
61	5.0	2.0	3.5	1.0	versicolor
62	5.9	3.0	4.2	1.5	versicolor
63	6.0	2.2	4.0	1.0	versicolor
64	6.1	2.9	4.7	1.4	versicolor
65	5.6	2.9	3.6	1.3	versicolor
66	6.7	3.1	4.4	1.4	versicolor
67	5.6	3.0	4.5	1.5	versicolor
68	5.8	2.7	4.1	1.0	versicolor

69	6.2	2.2	4.5	1.5	versicolor
70	5.6	2.5	3.9	1.1	versicolor
71	5.9	3.2	4.8	1.8	versicolor
72	6.1	2.8	4.0	1.3	versicolor
73	6.3	2.5	4.9	1.5	versicolor
74	6.1	2.8	4.7	1.2	versicolor
75	6.4	2.9	4.3	1.3	versicolor
76	6.6	3.0	4.4	1.4	versicolor
77	6.8	2.8	4.8	1.4	versicolor
78	6.7	3.0	5.0	1.7	versicolor
79	6.0	2.9	4.5	1.5	versicolor
80	5.7	2.6	3.5	1.0	versicolor
81	5.5	2.4	3.8	1.1	versicolor
82	5.5	2.4	3.7	1.0	versicolor
83	5.8	2.7	3.9	1.2	versicolor
84	6.0	2.7	5.1	1.6	versicolor
85	5.4	3.0	4.5	1.5	versicolor
86	6.0	3.4	4.5	1.6	versicolor
87	6.7	3.1	4.7	1.5	versicolor
88	6.3	2.3	4.4	1.3	versicolor
89	5.6	3.0	4.1	1.3	versicolor
90	5.5	2.5	4.0	1.3	versicolor
91	5.5	2.6	4.4	1.2	versicolor
92	6.1	3.0	4.6	1.4	versicolor
93	5.8	2.6	4.0	1.2	versicolor
94	5.0	2.3	3.3	1.0	versicolor
95	5.6	2.7	4.2	1.3	versicolor
96	5.7	3.0	4.2	1.2	versicolor
97	5.7	2.9	4.2	1.3	versicolor
98	6.2	2.9	4.3	1.3	versicolor
99	5.1	2.5	3.0	1.1	versicolor
100	5.7	2.8	4.1	1.3	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
103	7.1	3.0	5.9	2.1	virginica
104	6.3	2.9	5.6	1.8	virginica
105	6.5	3.0	5.8	2.2	virginica
106	7.6	3.0	6.6	2.1	virginica
107	4.9	2.5	4.5	1.7	virginica
108	7.3	2.9	6.3	1.8	virginica

```
109          6.7         2.5          5.8         1.8  virginica
110          7.2         3.6          6.1         2.5  virginica
111          6.5         3.2          5.1         2.0  virginica
112          6.4         2.7          5.3         1.9  virginica
113          6.8         3.0          5.5         2.1  virginica
114          5.7         2.5          5.0         2.0  virginica
115          5.8         2.8          5.1         2.4  virginica
116          6.4         3.2          5.3         2.3  virginica
117          6.5         3.0          5.5         1.8  virginica
118          7.7         3.8          6.7         2.2  virginica
119          7.7         2.6          6.9         2.3  virginica
120          6.0         2.2          5.0         1.5  virginica
121          6.9         3.2          5.7         2.3  virginica
122          5.6         2.8          4.9         2.0  virginica
123          7.7         2.8          6.7         2.0  virginica
124          6.3         2.7          4.9         1.8  virginica
125          6.7         3.3          5.7         2.1  virginica
126          7.2         3.2          6.0         1.8  virginica
127          6.2         2.8          4.8         1.8  virginica
128          6.1         3.0          4.9         1.8  virginica
129          6.4         2.8          5.6         2.1  virginica
130          7.2         3.0          5.8         1.6  virginica
131          7.4         2.8          6.1         1.9  virginica
132          7.9         3.8          6.4         2.0  virginica
133          6.4         2.8          5.6         2.2  virginica
134          6.3         2.8          5.1         1.5  virginica
135          6.1         2.6          5.6         1.4  virginica
136          7.7         3.0          6.1         2.3  virginica
137          6.3         3.4          5.6         2.4  virginica
138          6.4         3.1          5.5         1.8  virginica
139          6.0         3.0          4.8         1.8  virginica
140          6.9         3.1          5.4         2.1  virginica
141          6.7         3.1          5.6         2.4  virginica
142          6.9         3.1          5.1         2.3  virginica
143          5.8         2.7          5.1         1.9  virginica
144          6.8         3.2          5.9         2.3  virginica
145          6.7         3.3          5.7         2.5  virginica
146          6.7         3.0          5.2         2.3  virginica
147          6.3         2.5          5.0         1.9  virginica
148          6.5         3.0          5.2         2.0  virginica

149          6.2         3.4          5.4         2.3  virginica
150          5.9         3.0          5.1         1.8  virginica
```

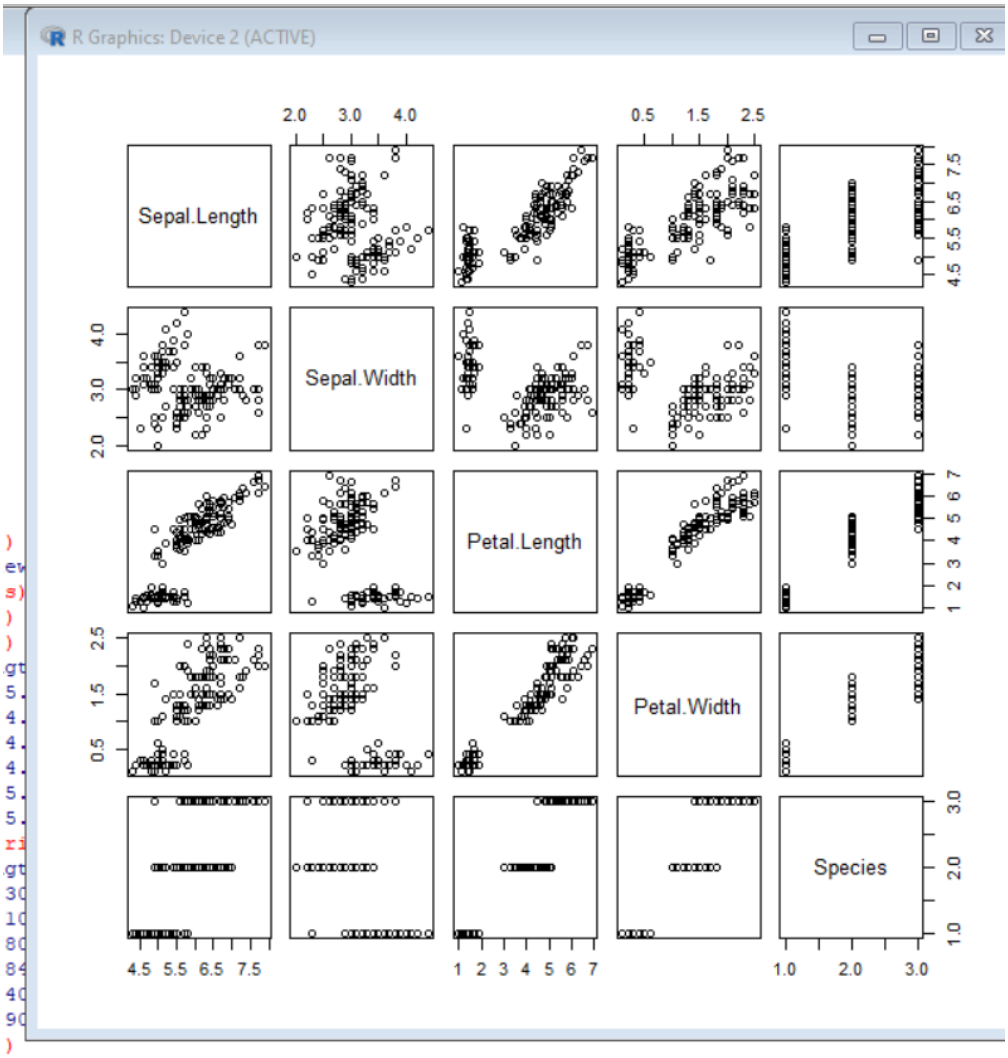
Plot the iris dataset

```
> view(iris)
Error in view(iris) : could not find function "view"
> #view(iris)
> 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          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> plot(iris)
> plot(iris[,3:4])
```

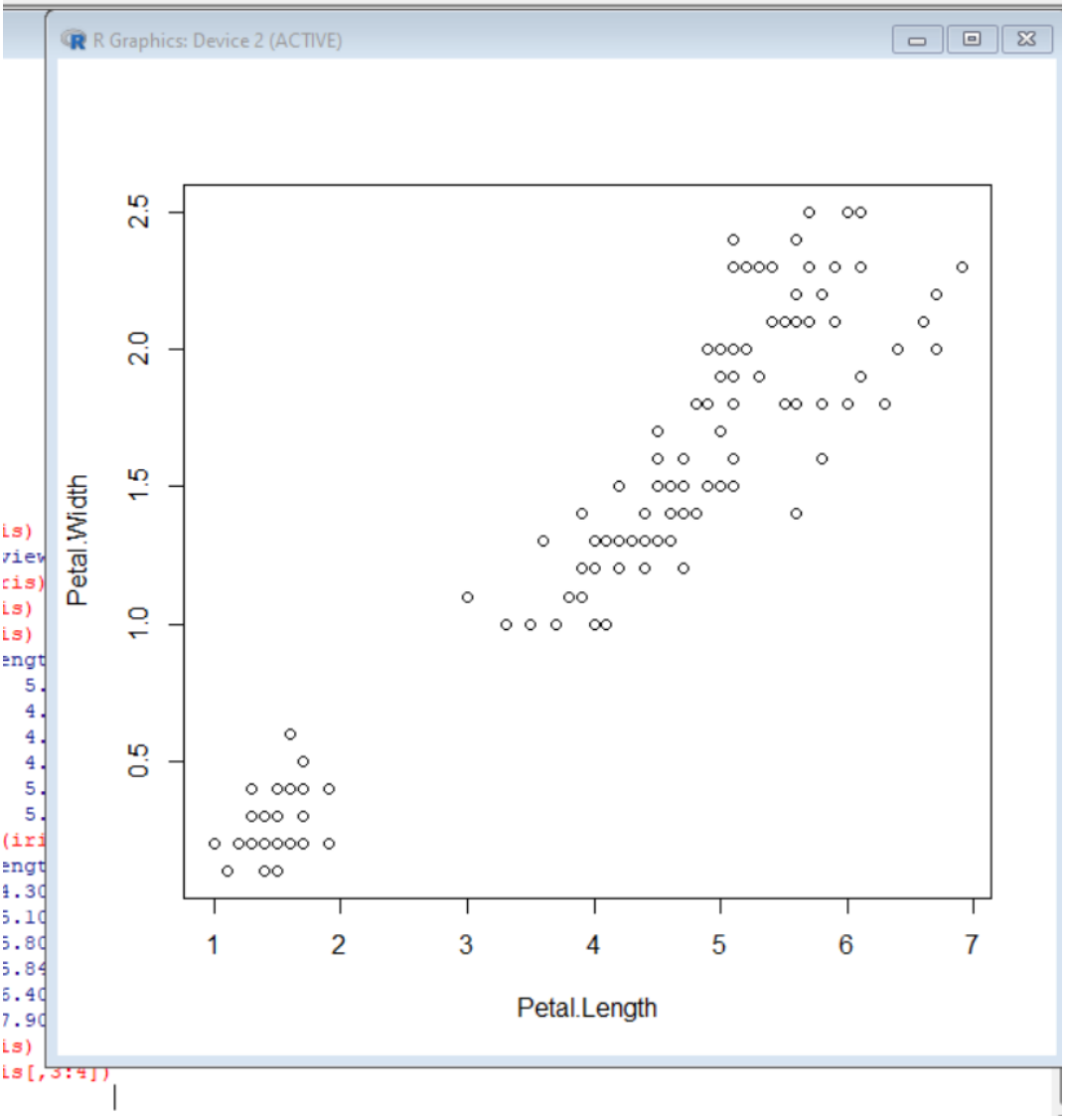
327 Mohit Kapoor

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



327 Mohit Kapoor



Kmeans clustering

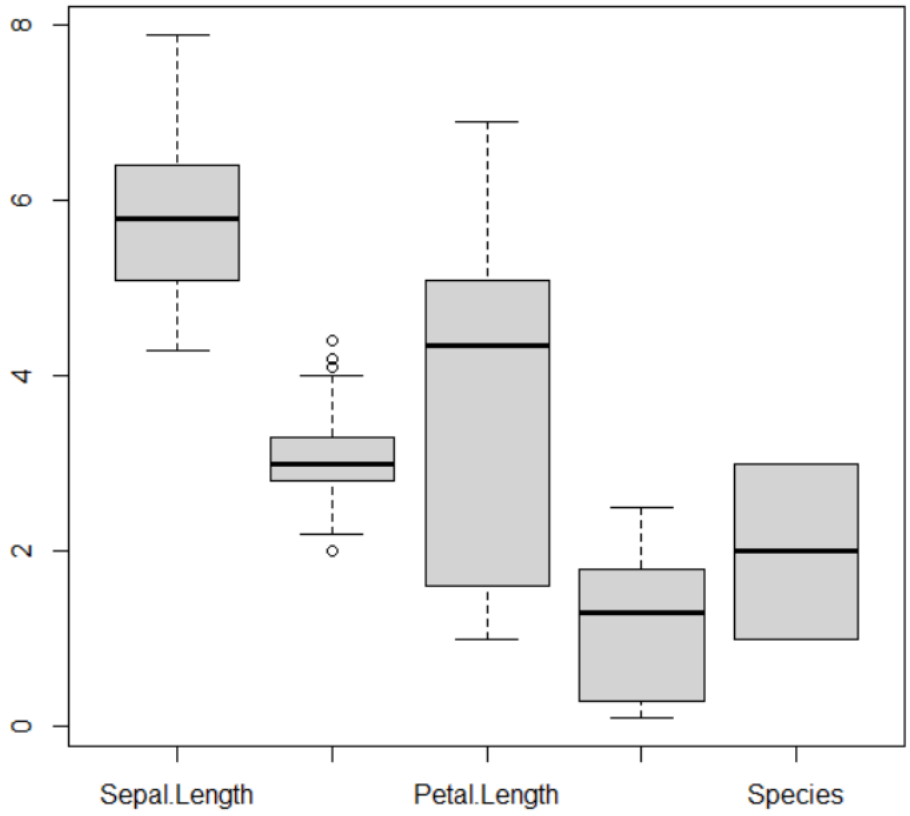
```
> kmeanscl=kmeans(iris[,3:4],3)
> kmeanscl
K-means clustering with 3 clusters of sizes 48, 50, 52

Cluster means:
  Petal.Length Petal.Width
1     5.595833    2.037500
2     1.462000    0.246000
3     4.269231    1.342308

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [43] 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1
 [85] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1
[127] 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 16.29167  2.02200 13.05769
 (between_SS / total_SS =  94.3 %)

Available components:

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

327 Mohit Kapoor
Print confusion matrix

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

    setosa versicolor virginica
1       0          2        46
2      50          0         0
3       0         48         4
```

CALCULATION OF ACCURACY 94.3%

Boxplot the iris dataframe



Conclusion: Clustering or cluster analysis represents one of the most important tasks of data analysis. It essentially uncovers groups (so-called clusters) in un labeled data – with elements in the same group sharing similar values of the dataset's features. Clustering belongs to the group of unsupervised machine learning problems. It can be formulated as an optimization task which in many of its forms is known to be NP-hard.

Practical 9

Aim: Demonstration of Time-Series Forecasting

Theory: Time series forecasting occurs when you make scientific predictions based on historical time stamped data. It involves building models through historical analysis and using them to make observations and drive future strategic decision-making. An important distinction in forecasting is that at the time of the work, the future outcome is completely unavailable and can only be estimated through careful analysis and evidence-based priors.

Time series forecasting is the process of analyzing time series data using statistics and modeling to make predictions and inform strategic decision-making. It's not always an exact prediction, and likelihood of forecasts can vary wildly—especially when dealing with the commonly fluctuating variables in time series data as well as factors outside our control. However, forecasting insight about which outcomes are more likely—or less likely—to occur than other potential outcomes. Often, the more comprehensive the data we have, the more accurate the forecasts can be. While forecasting and "prediction" generally mean the same thing, there is a notable distinction. In some industries, forecasting might refer to data at a specific future point in time, while prediction refers to future data in general. Series forecasting is often used in conjunction with time series analysis. Time series analysis involves developing models to gain an understanding of the data to understand the underlying causes. Analysis can provide the "why" behind the outcomes you are seeing. Forecasting then takes the next step of what to do with that knowledge and the predictable extrapolations of what might happen in the future.

Code:

Import required packages

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

Import time series data: Airline passenger traffic

[2]:
```
data = pd.read_csv('C:\Users\Mohit Kapoor\Downloads\airline-passenger-traffic(1).csv')
data.columns = ['Month','Passengers']
data['Month'] = pd.to_datetime(data['Month'], format='%Y-%m')
data = data.set_index('Month')
data.head()
```

[2]:

Month	Passengers
1949-02-01	118.0
1949-03-01	132.0
1949-04-01	129.0
1949-05-01	121.0
1949-06-01	135.0

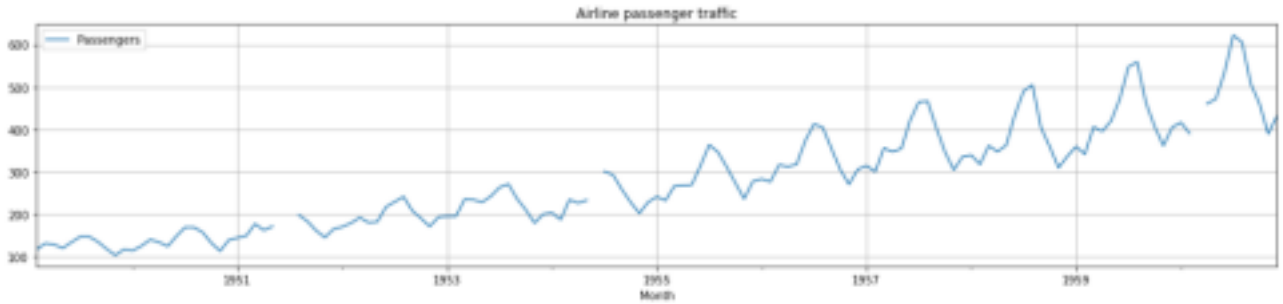
Time series analysis

Plot time series data

327 Mohit Kapoor

[3]:

```
data.plot(figsize=(20, 4))
plt.grid()
plt.legend(loc='best')
plt.title('Airline passenger traffic')
plt.show(block=False)
```



Missing value treatment

Linear interpolation

[4]:
```
data = data.assign(Passengers_Linear_Interpolation=data.Passengers.interpolate(method='linear'))
data[['Passengers_Linear_Interpolation']].plot(figsize=(20, 4))
plt.grid()
plt.legend(loc='best')
plt.title('Airline passenger traffic: Linear interpolation')
plt.show(block=False)
```

Use linear interpolation to impute missing values

327 Mohit Kapoor

[5]:
```
data['Passengers'] = data['Passengers_Linear_Interpolation']
data.drop(columns=['Passengers_Linear_Interpolation'],inplace=True)
data.head()
```

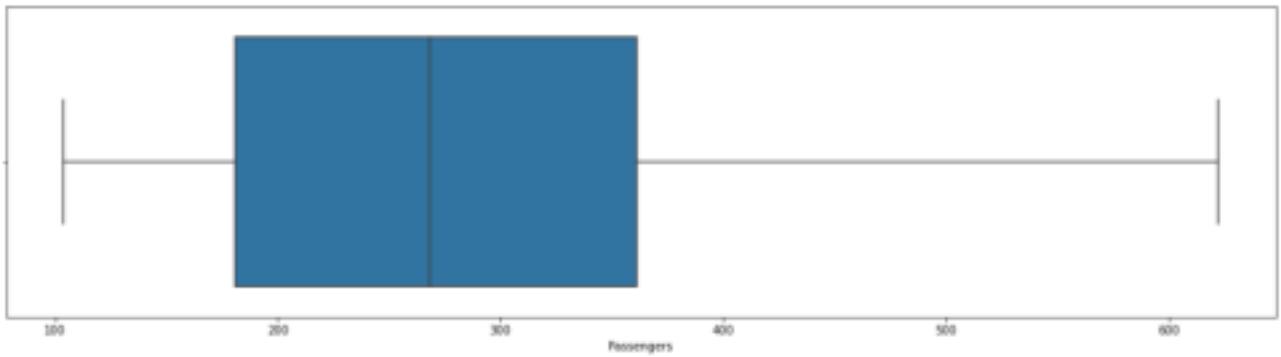
[5]:

Month	Passengers
1949-02-01	118.0
1949-03-01	132.0
1949-04-01	129.0
1949-05-01	121.0
1949-06-01	135.0

Outlier detection

Box plot and interquartile range

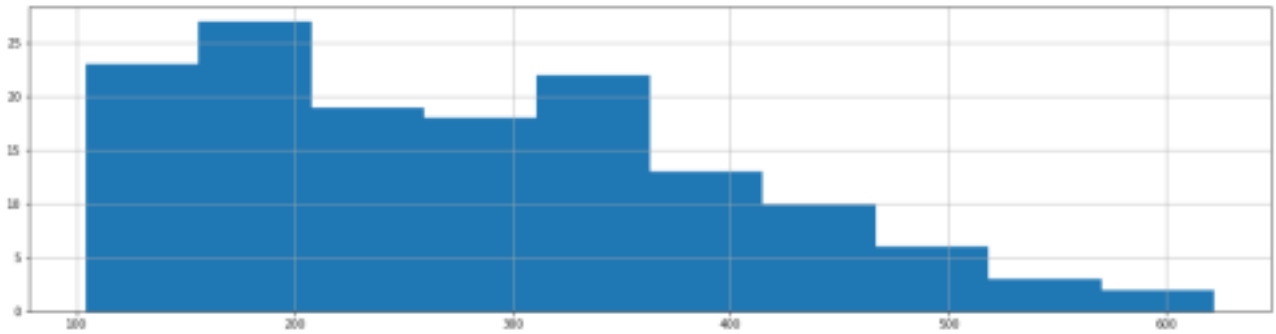
[6]:
```
import seaborn as sns
fig = plt.subplots(figsize=(20, 5))
ax = sns.boxplot(x=data['Passengers'],whis=1.5)
```



Histogram plot

327 Mohit Kapoor

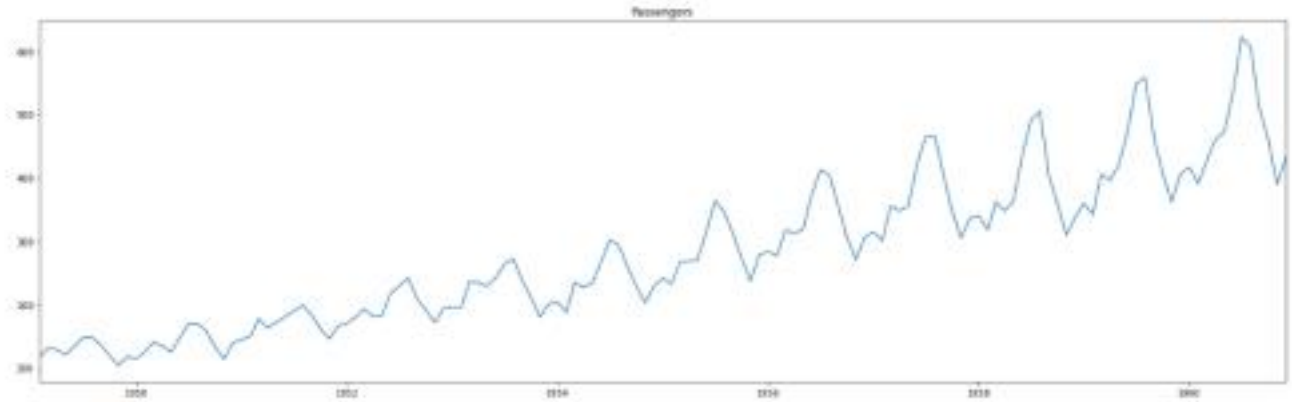
[7]:
```
fig = data.Passengers.hist(figsize = (20,5))
```



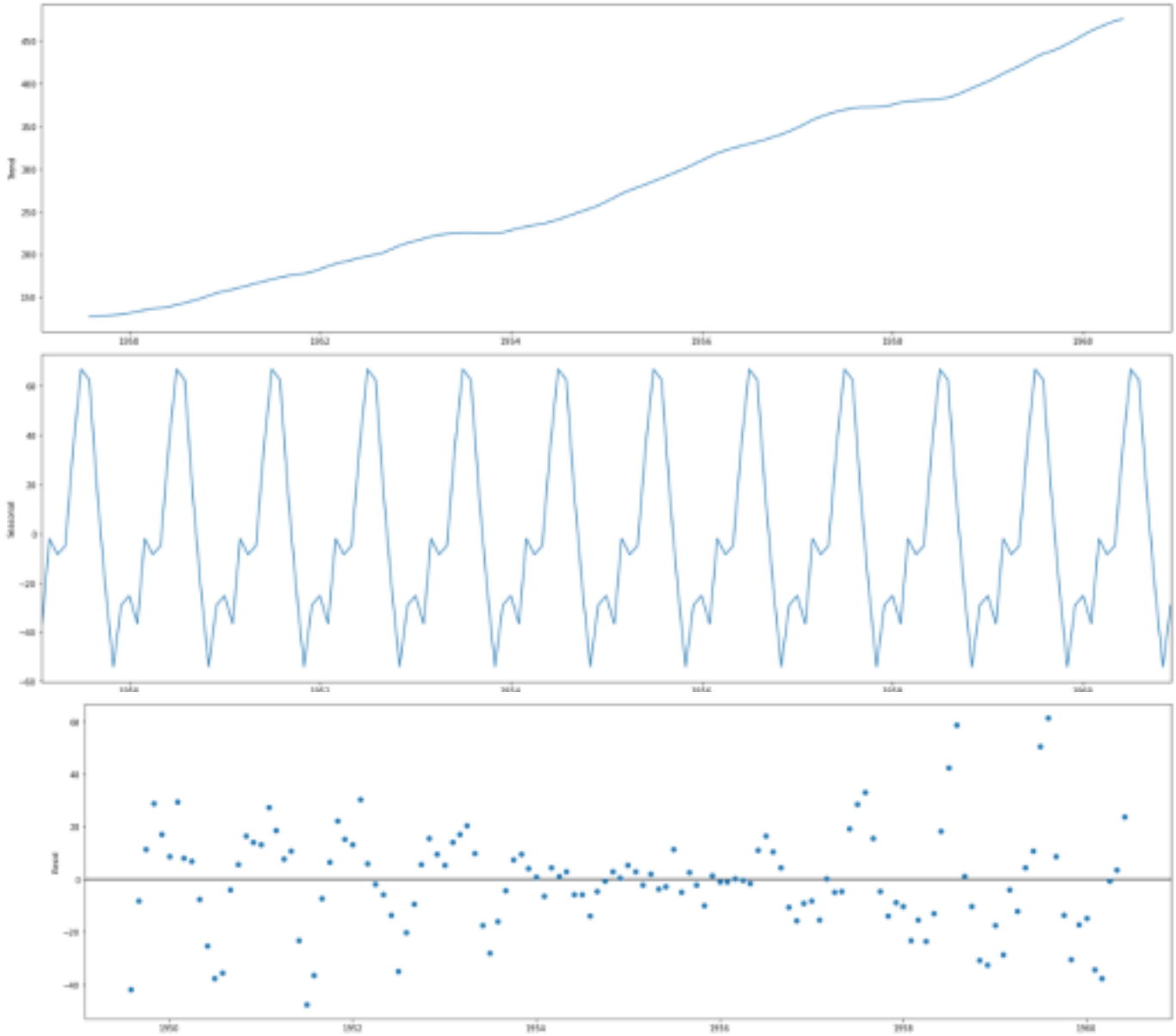
Time series Decomposition

Additive seasonal decomposition

[8]:
```
from pylab import rcParams
import statsmodels.api as sm
rcParams['figure.figsize'] = 20,24
decomposition = sm.tsa.seasonal_decompose(data.Passengers, model='additive') # additive seasonal index
fig = decomposition.plot()
plt.show()
```



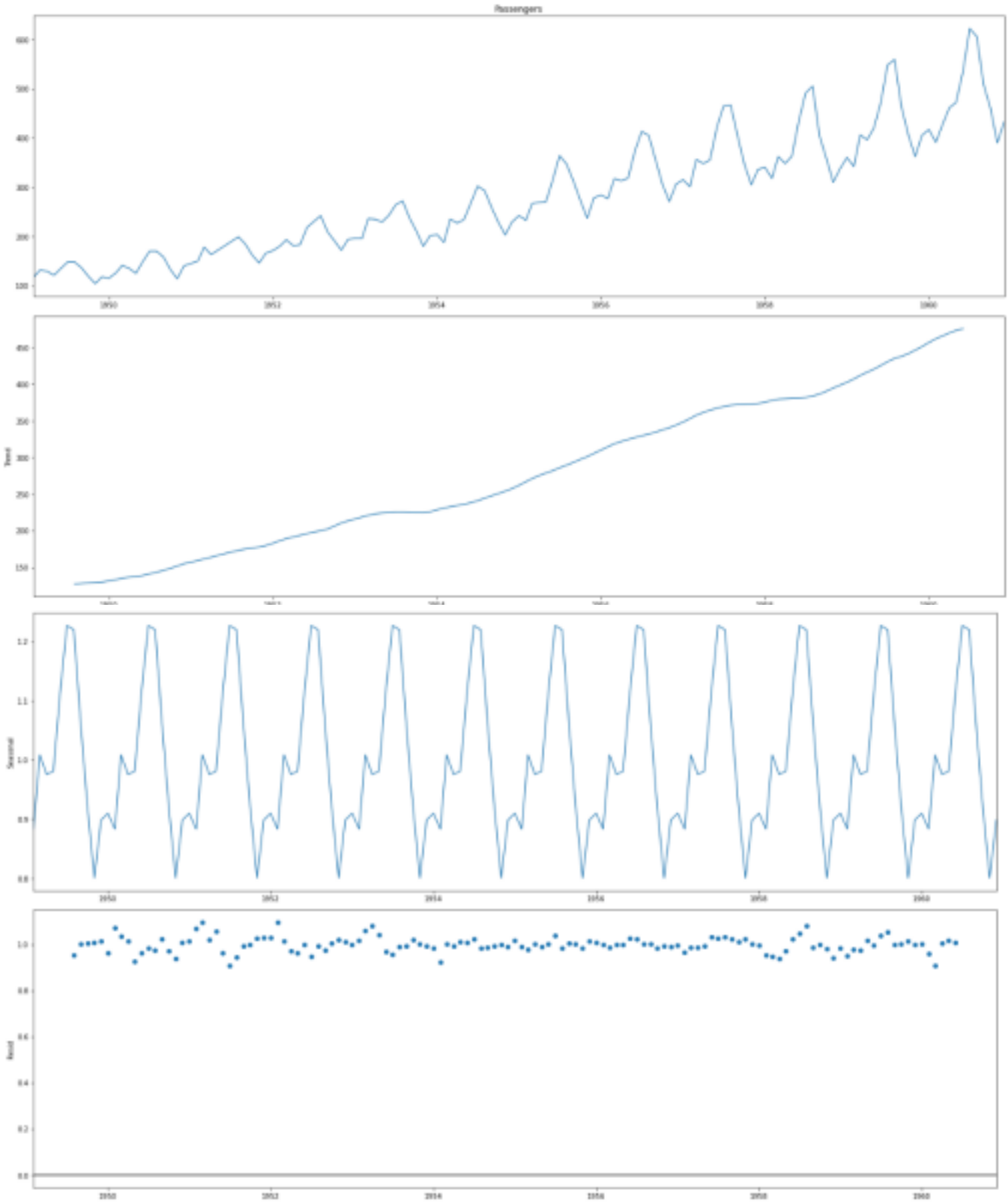
327 Mohit Kapoor



Multiplicative seasonal decomposition

[9]:
```
decomposition = sm.tsa.seasonal_decompose(data.Passengers, model='multiplicative') # multiplicative seasons
fig = decomposition.plot()
plt.show()
```

327 Mohit Kapoor


Passengers
Trend
Seasonal
Resid

Build and evaluate time series forecast

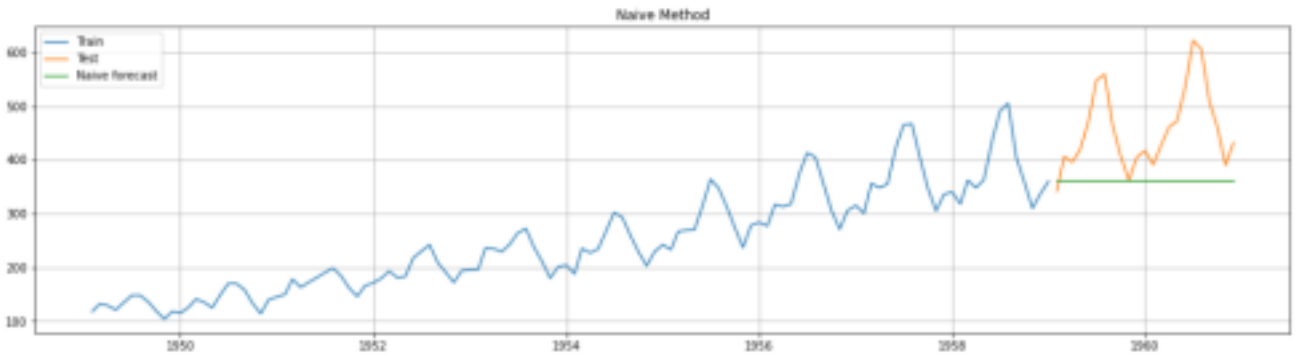
Split time series data into training and test set

[10]:
```
train_len = 120
train = data[0:train_len] # first 120 months as training set
test = data[train_len:] # last 24 months as out-of-time test set
```

[11]:
```
y_hat_naive = test.copy()
y_hat_naive['naive_forecast'] = train['Passengers'][train_len-1]
```

Plot train, test and forecast

[12]:
```
plt.figure(figsize=(20,5))
plt.grid()
plt.plot(train['Passengers'], label='Train')
plt.plot(test['Passengers'], label='Test')
plt.plot(y_hat_naive['naive_forecast'], label='Naive forecast')
plt.legend(loc='best')
plt.title('Naive Method')
plt.show()
```



Calculate RMSE and MAPE

[13]:
```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(test['Passengers'], y_hat_naive['naive_forecast'])).round(2)
mape = np.round(np.mean(np.abs(test['Passengers']-y_hat_naive['naive_forecast'])/test['Passengers'])*100,2)

results = pd.DataFrame({'Method':['Naive method'], 'MAPE': [mape], 'RMSE': [rmse]})
results = results[['Method', 'RMSE', 'MAPE']]
results
```

[13]:

	Method	RMSE	MAPE
0	Naive method	121.4	19.68

Conclusion: Time Series Forecasting is an integral part of machine learning which is used in many applications such as weather forecasting, stock prices forecasting, resource allocation, business planning, and a lot more. Time Series Forecasting a bit hard to handle because it is directly related to the time component so that it may result in a lot of

prediction problems.

Time series forecasting is very important for many application domains. The success rate of predicting the future is not easy to calculate because of its direct dependency on time constrain. Future predictions always have chances of errors. A lot of active research works are going on on this subject during several years. Many important models have been proposed for improving the accuracy and efficiency of time series modelling and forecasting. Time series forecasting always have room for improvement in order to yield more accurate data

Practical 10

Aim: Use of R Markdown and RStudio Cloud

Theory: R Markdown is a file format for making dynamic documents with R. An R Markdown document is written in markdown (an easy-to-write plain text format) and contains chunks of embedded R code

R Markdown files are designed to be used with the rmarkdown package. rmarkdown comes installed with the RStudio IDE, but you can acquire your own copy of rmarkdown from CRAN with the command install.packages('rmarkdown')

R Markdown files are the source code for rich, reproducible documents. You can transform an R Markdown file in two ways.

1. **knit** - You can *knit* the file. The rmarkdown package will call the knitr package. knitr will run each chunk of R code in the document and append the results of the code to the document next to the code chunk. This workflow saves time and facilitates reproducible reports.

Consider how authors typically include graphs (or tables, or numbers) in a report. The author makes the graph, saves it as a file, and then copy and pastes it into the final report. This process relies on manual labor. If the data changes, the author must repeat the entire process to update the graph.

In the R Markdown paradigm, each report contains the code it needs to make its own graphs, tables, numbers, etc. The author can automatically update the report by re-knitting.

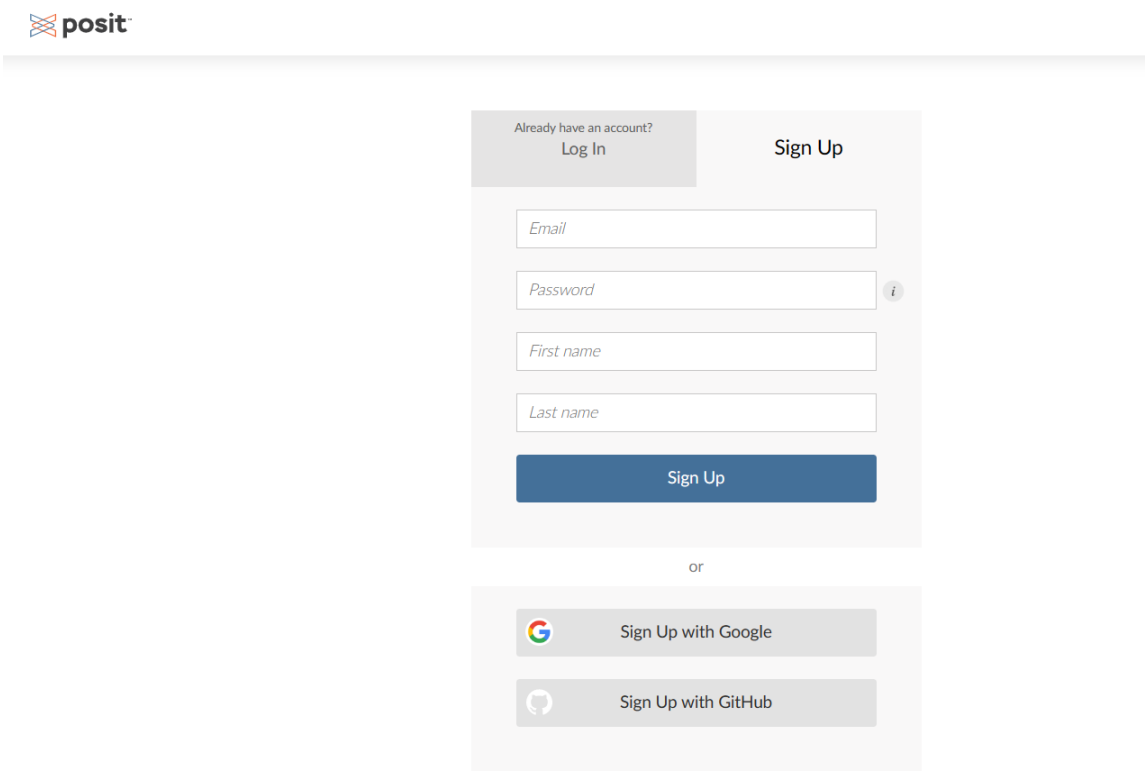
2. **convert** - You can *convert* the file. The rmarkdown package will use the pandoc program to transform the file into a new format. For example, you can convert your .Rmd file into an HTML, PDF, or Microsoft Word file. You can even turn the file into an HTML5 or PDF slideshow. rmarkdown will preserve the text, code results, and formatting contained in your original .Rmd file.

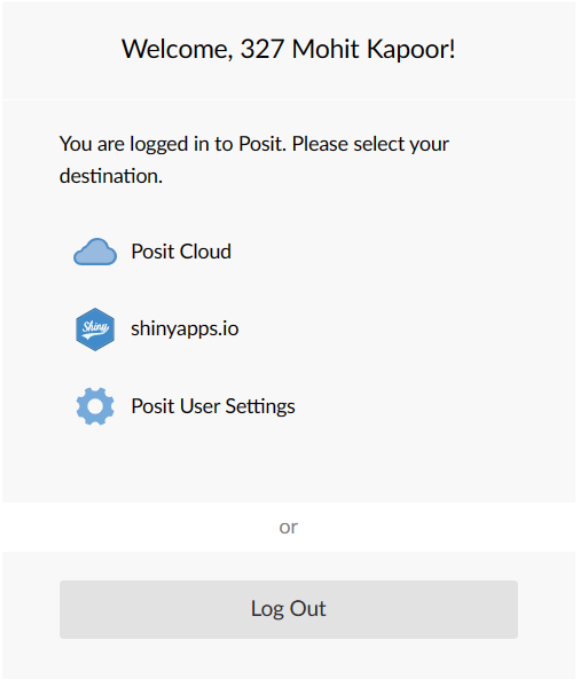
Posit Cloud is a lightweight, cloud-based solution that allows anyone to do, share, teach and learn data science online.

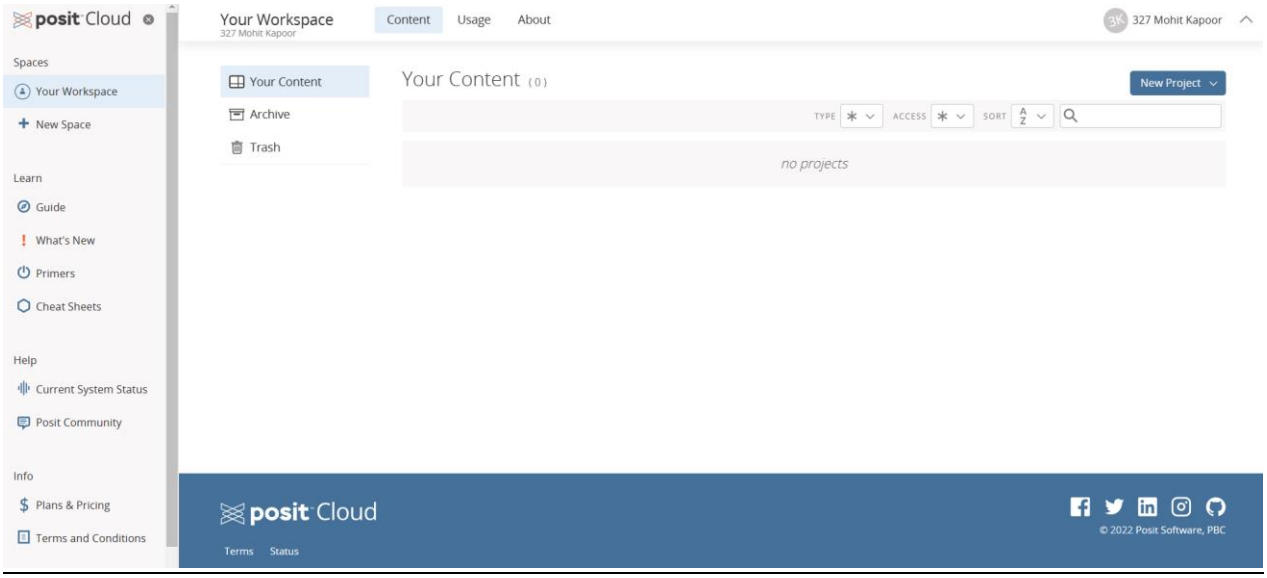
There is nothing to configure and no dedicated hardware, installation or annual purchase contract required. Individual users, instructors and students only need a browser to do, share, teach and learn data science.

Code:

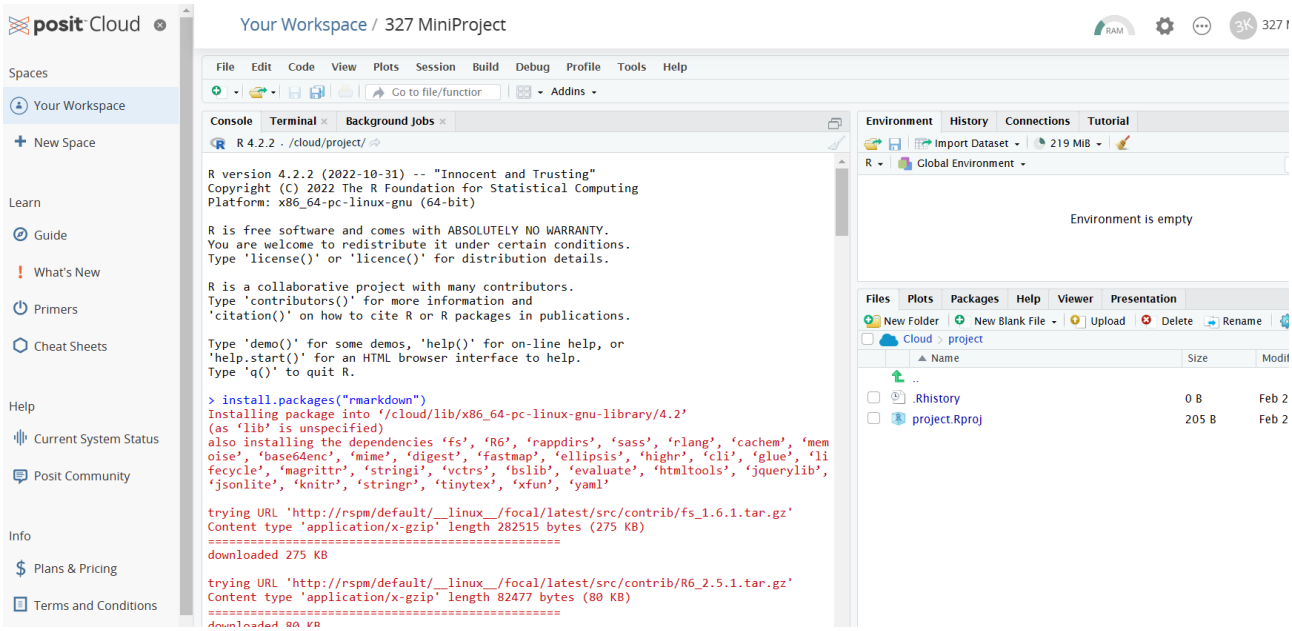
Login or Register your account into Posit Cloud (formerly knows as RStudio Cloud)



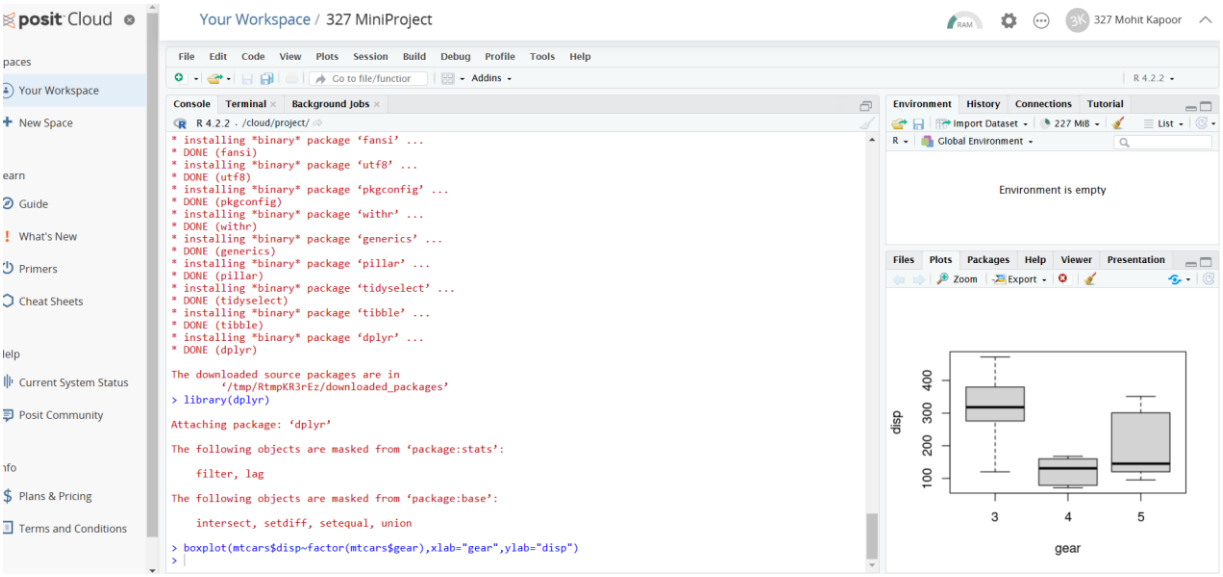


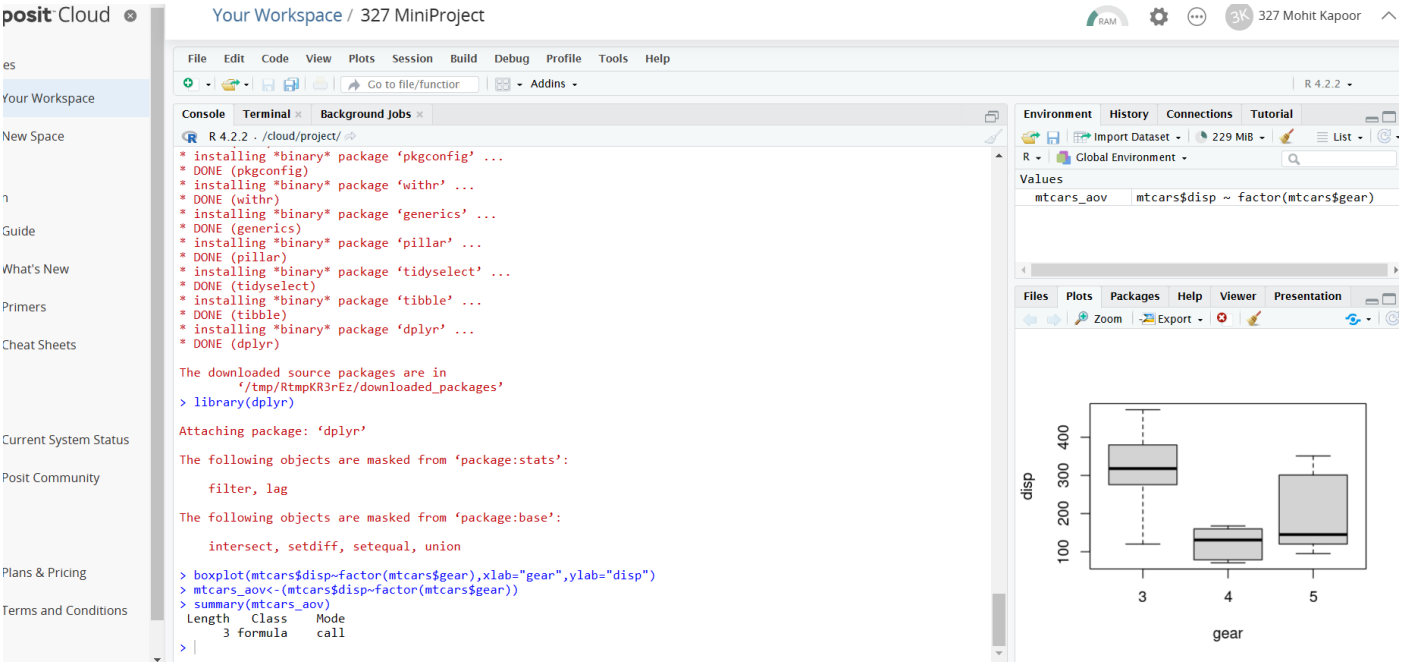


Installing RMarkdown in RStudio Cloud



Creating a Mini Project using R markdown





Conclusion: Posit Cloud supports the hosting of a variety of data products, including Shiny, Dash, Streamlit, Flask and Bokeh apps, as well as some flavors of R Markdown and Quarto documents. The cloud can be used for implement big as well as mini projects of R language coding in this cloud. This is entirely online which is hassle-free in terms of organizing a large amount of programming files.