Data Science

Assignment - II ...-

P. Siva Nagini

Reg No :- 22307219

Group :- I st BSC [DSCS]

Sem - II

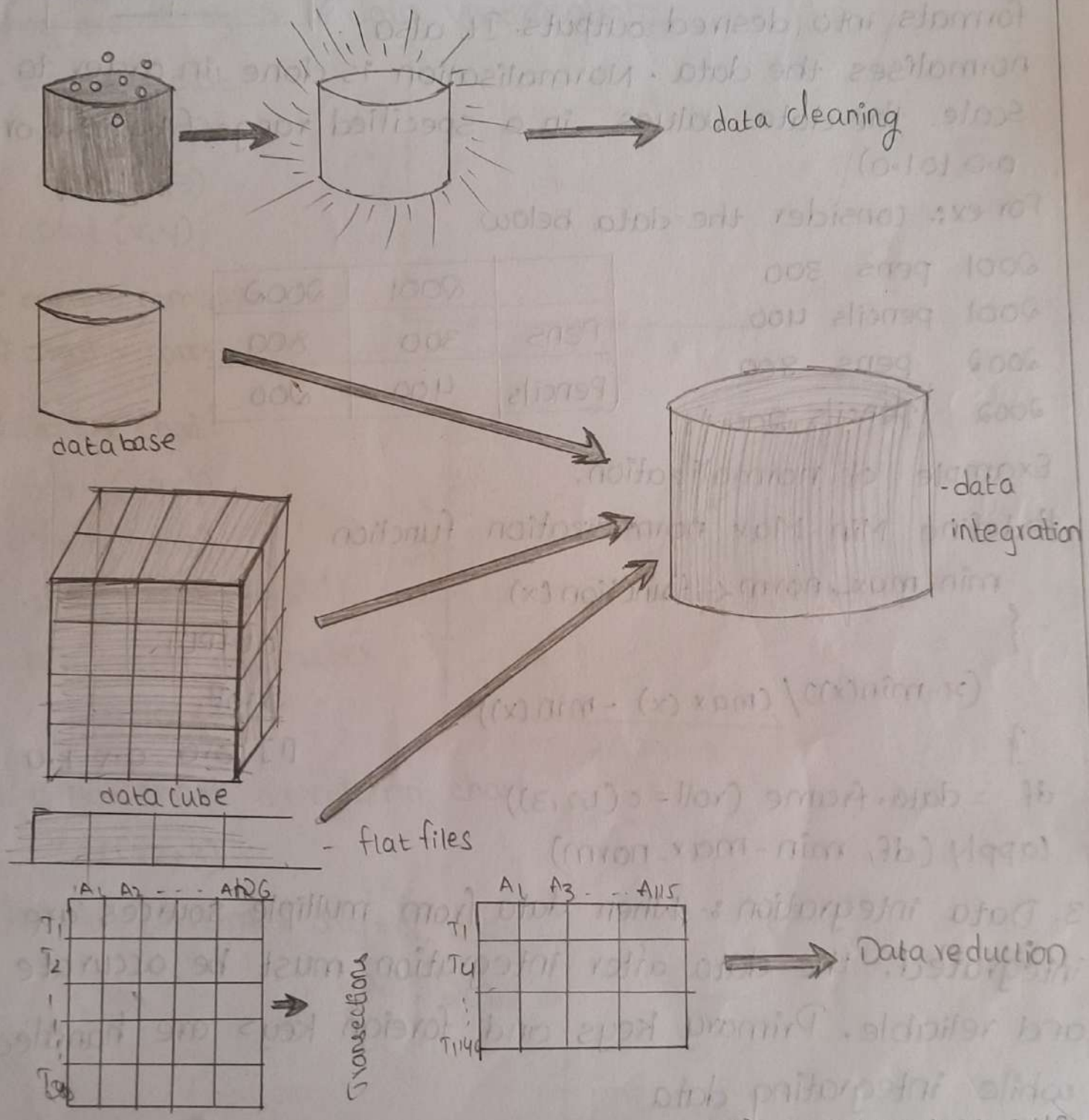
1. What is data preprocessing & What are the steps involved?

A Data preprocessing :- It is used to transform the raw data in useful and efficient format

They are 4 steps:-

1. Data cleaning 2. Data integration 3. Data reduction

4. Data transformation.



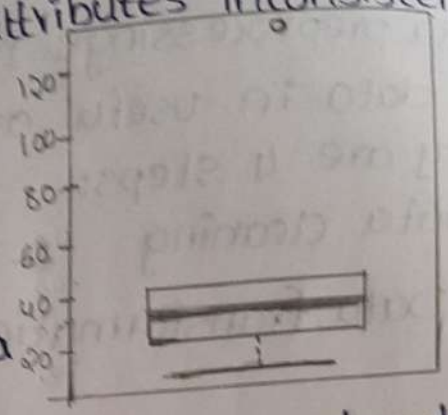
Data transformation :- -2, 32, 100, 59, 48 → -0.02, 0.32, 1.00, 0.50, 0.48

Forms of data preprocessing:-

1. Data cleaning :- It handles missing values NULL or unwanted values, duplicate values missspelt attributes inconsistent data types and handling outliers.

Detect outliers with boxplot function:

```
df = data.frame (roll = c(10,20,30,40,123))
boxplot (df)
```



2. Data transformation :- turns raw data formats into desired outputs. It also normalises the data. Normalisation is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

For ex:- consider the data below

2001 pens 300
2001 pencils 400
2002 pens 800
2002 pencils 200

	2001	2002
Pens	300	800
Pencils	400	200

Example of normalization.

```
#define Min-Max normalization function
  min-max_norm <- function (x)
  {
    (x-min(x))/ (max (x) - min (x))
  }
df = data.frame (roll = c (1,2,3))
lapply (df, min-max_norm)
```

OUTPUT

```
$roll
[1] 0.0 0.5 1.0
```

3. Data integration :- When data from multiple sources are integrated, the data after integration must be accurate and reliable. Primary keys and foreign keys are handled while integrating data

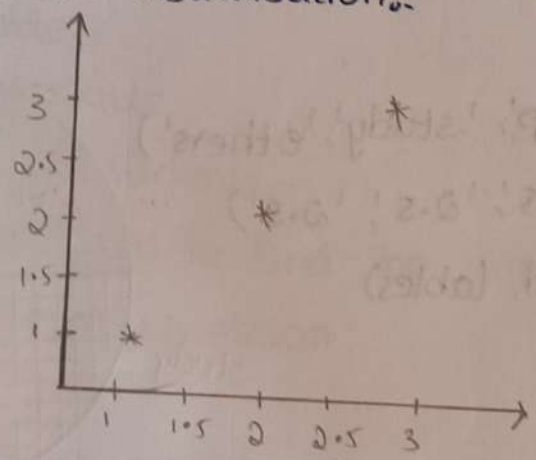
4. Data reduction:- Data reduction is a process that redu the volume of original data and represents it in a much smaller volume. Data reduction ten techniques are used obtain a reduced representation of the dataset that is much smaller in volume by maintaining the integrity of original data.

2. How is R used in data visualisation?

A They are 6 types of data visualisation:-

1. #scatter plot

```
x = c(1,2,3)
y = c(1,2,3)
plot (x,y)
```



c means combine

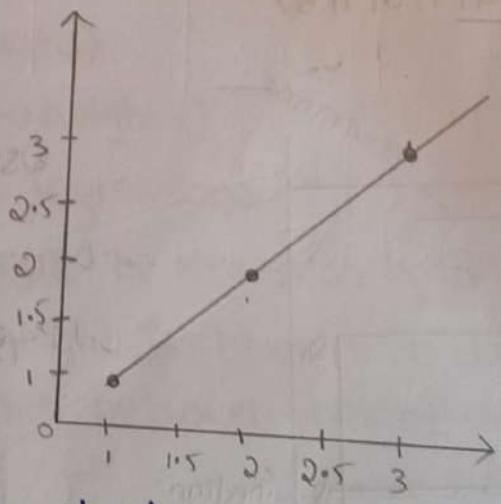
means commends.

2. #line chart

```
x = c(1,2,3)
y = c(1,2,3)
plot (x,y, type='l')
```

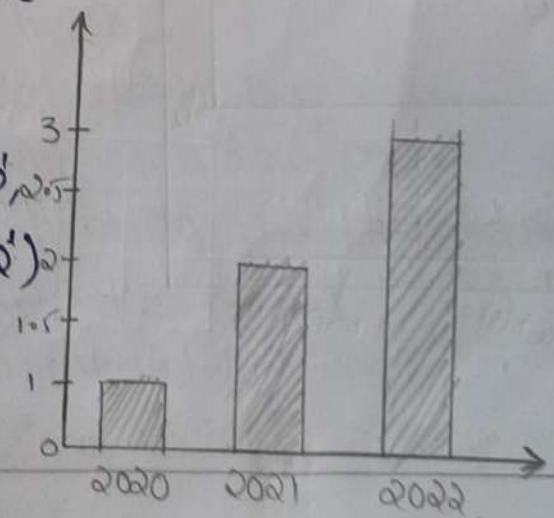
Pch - point character

x,y tables.



3. # Bar chart or column chart.

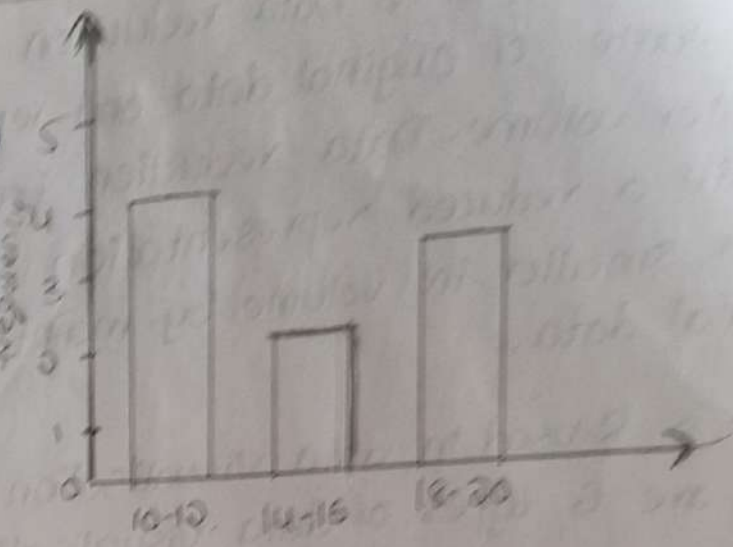
```
x = c(1,2,3)
barplot (x, names arg = c("2020",
                  '2021', '2022'))
```



4. # histogram

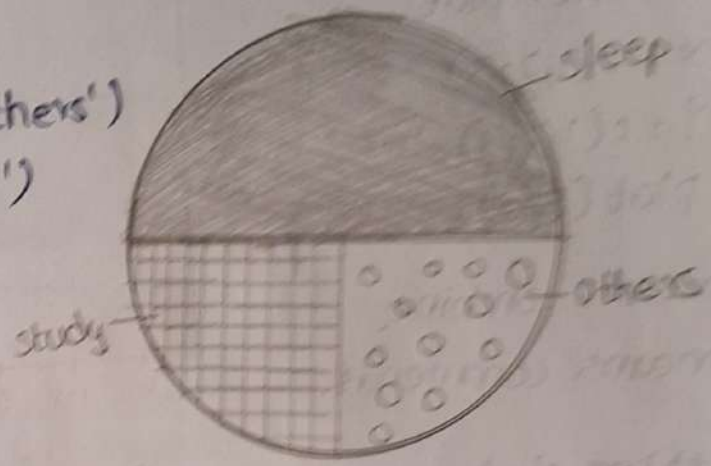
```
z = c(15,15,20,20,20,10,10,10,10)
hist (z)
```

To summarize discrete or continuous data that are measured on an interval scale.



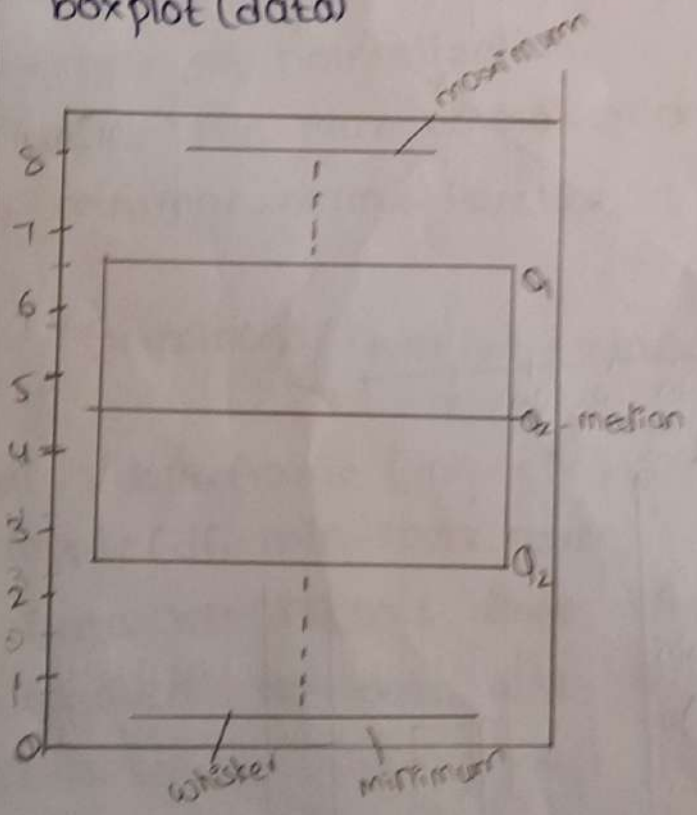
5. # pie chart

```
lables = c('sleep', 'study', 'others')
hours spend = c('5', '2.5', '2.5')
pie (hours spend, lables)
```



6. # box plot

```
data = c(1,2,3,4,5,6,7,8)
boxplot (data)
```



25% of data is less than 2.5 - 1st quartile

75% of data is less than 6.5 - 3rd quartile

Off of the data is less than 4.5 - 2nd quartile.

3. Explain in detail about linear regression and multiple linear regression

A. linear regression:- It is used to establish a relationship b/w two variables. One is dependent variable. Another variable is independent variable.

In linear regression these two variables are related through an equation. $\boxed{Y = ax + b}$

Y is dependent variable

x is independent variable

a & b are constants & coefficients.

In R language lm() is used to find the coefficients for linear and multiple linear regression.

Syntax:-

lm() stands for linear model

model ← lm(formula, data)

Ex: model ← lm(weight y ~ height x)

Steps to establish linear regression:-

A simple example of regression is to predict the weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

Here Y is weight, & x is height.

The steps to create the relationship is:-

1. Gather the height and weight of a few people.
2. Create a relationship model using the lm() in R
3. Find the coefficients from the model
4. know the average error in prediction. Also called residual
5. Use the predict () to predict the weight of new person

For ex:-

```
heightx = c(1,2,3)
weighty = c(1,3,4)
relation = lm(weighty~heightx)
print(relation)
```

o/p:-

b = -0.3333

a = 1.5

Hence the line eqn in

$Y = 1.5x - 0.33$

To predict the weight of new person whose height is 2.5

```
newdata <- data.frame (heightx=2.5)
result <- predict (relation, newdata)
print (result)
```

o/p:-

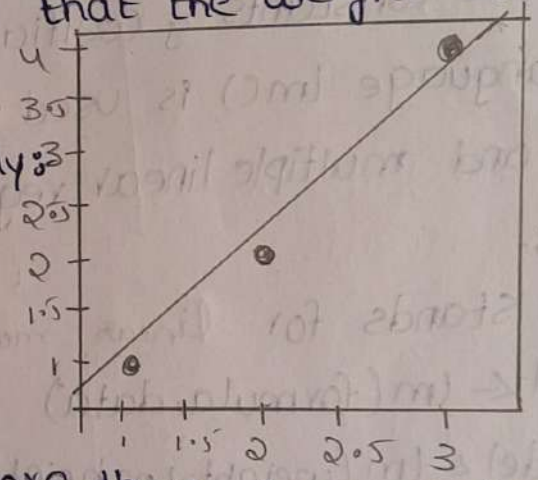
1

3.41

The regression model predicted that the weight of the person is 3.41

Visualize the regression graphically:

```
plot (heightx, weighty)
abline (lm(weighty~heightx))
```



Multiple linear regression:- It finds relationship between more than two variables. In multiple regression we have more than one independent variable.

The general mathematical equation for multiple regression is

$$\boxed{Y = a_1x_1 + a_2x_2 + \cdots + b}$$

Here

Y is the dependent variable

$x_1, x_2 \ldots x_n$ are independent variable

$b, a_1, a_2 \ldots a_n$ are the coefficients.

Syntax:-

The basic syntax for lm() in multiple regression is

lm (weighty ~ height x + age x_2)

Ex:-

height x_1 = c(1,2,3)

weighty = c(1,3,4)

age x_2 = c(0,3,4)

relation = lm (weighty ~ height x1 + age x_2)

o/p:-

lm(formulae =

weighty ~ height x + age x_2)

coefficients:

	height x_1	age x_2
0.5	0.5	0.5

4. Explain how to handle packages in R.

A. R packages are collection of functions, data and documentation that extend the functionality of R.

They are three types of handling packages in R:-

1. Installing packages:-

To install a package you can use the install. packages () function.

Ex:- install. packages ("dplyr") will install the "dplyr" package.

You only need to install a package once unless it is updated ~~a~~ new to a new version.

2. checking installed packages:-

To check which packages are installed on your system, yo can use the installed.packages()

Explain

Ex:- installed.packages() will list all the packages installed on your system. You can also use the find.package() function to check if a specified package is installed.

Ex:- find.package("dplyr") will return the path to the "dplyr" package if it is installed or an empty string if it is not installed.

3 loading packages:-

Once a package is installed, you need to load it into your R session to use its functions. You can use the library() function to load a package.

Ex:- library(dplyr) will load the "dplyr" package.

You can also use require() function instead

Types of learning :-

1. Supervised learning :- This machine learning type got its name because the machine is "supervised" while it's learning. Here we provide training data with class labels. The model learns the relationship between the features and class from training data.

After the model is trained, we can use the tal models to predict the class of new data.

Ex :- 1. Predicting real estate prices

2. Classifying whether bank transections are fradulent or not.

3. Finding disease risk factors.

4. Determining whether loan applicats are low-risk or high-risk.

5. Predicting the failure of industrial equipment's mechanical parts.

common algorithms used during supervised learning include neural networks, decision trees, linear regression, and support vector machines.

2. Unsupervised learning :- This machine learning type is very helpful when you need to identify patterns common algorithms used in unsupervised learning include hidden Markov models, k-means, hierarchical Clustering and Gaussian Mixture models.

common applications also include clustering. Clustering groups data based on specific properties. These groups are called clusters. It indentifies the

rules existing between the clusters.

Ex:- Creating customer groups based on purchase behaviour.

Reinforcement learning:-

It is the lowest to how human learn, It means by interacting with its environment. It also gets a positive reward for correct and negative reward for incorrect common algorithms include temporal difference, deep adversarial networks and Q-learning.

Ex:- If the algorithm classifies them as high-risk and They default, the algorithm gets a positive reward. If they don't default, the algorithm gets a negative reward.

In the end both instances help the machine learn by understanding both the problem and environment better.

1. Teaching cars to park themselves and drive.
2. Dynamically controlling traffic lights to reduce traffic jams.
3. Training robots to learn policies using raw video images as input that they can use to replicate the action they see.

5. Explain about factors.

A. Factors :- Factors help organise data that falls into specific groups. Factor have levels. These levels are unique labels.

Factor also allow due to specify the order of the levels.

Program:-

```
a = factor(c("high", "low", "low", "low", "high"))
print(a)
```

o/p.

[1] ~~how~~
high low
low low
high.

levels :- high, low.

creating a vector

```
a = factore (c("high", "low", "low", "low", "high", "high", "high")
levels = c("low", "high"), ordered = TRUE)
print (sum(a > "low")) or print(a)
```

o/p:-

[1] 4

6. What is the use of summary command in R

A. The str() c:

A. The str() command shows the structure of data. It will inform you about the number of rows and columns in the do table data values in the columns with their respective heads.

The summary() command will provide you with a statistical summary of your data, the

It gives the output as the ~~al~~ largest value in data, the least value or mean and median and another similar type of information.

It gives mean, median, 1st quartile, 3rd quartile, min, max.

Ex:-

```
data = c(12,7,5)
summary(data)
```

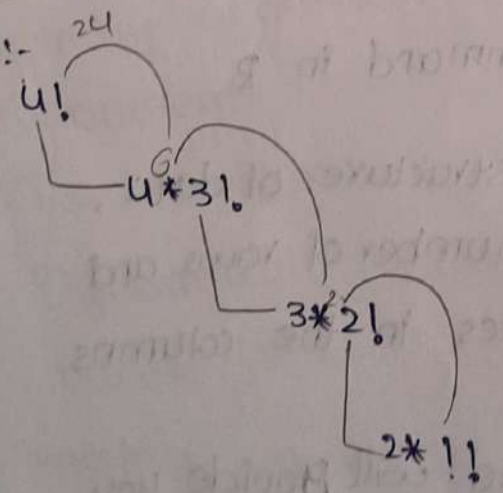
O/P:-

min	1st Qua	median	mean	3rd quar	Max
5	6	7	8	9.5	12

7. What is a recursive function? Explain.

A- Recurssion:- Recurssion function is a function that calls itself.

Ex:-




```
function
factorial(n) {
    1   if n==0 or n==1
    n* factorial(n-1)   n>1
}
```

Write a program to find factorial of given number using recursion

```
factorial ← function (n)
{
  if (n==0||n==1)
    return (1)
  else
    return (n* factorial (n-1))
}
```

O/P

24

<u>function call</u>

factorial (4)

Recurssion stack.

factorial (1) if (1==0||1==1) ✓ return 1

factorial (2)
if (2==0||2==1) ✗
else
return 2* factorial (1)

factorial (3)
if (3==0||3==1) ✗
else
return 3* factorial (2)

factorial (4)
if (4==0||4==1) ✗
else
return 4* factorial (3)

O/P

24

8. Explain about F- statistics.

A. F-statistics or F-tests are statistical measures used to compare the variance b/w groups. F-statistics using the anova () function or the summary () function applied to linear regression models. The var.test () function calculates an F-test to compare the variances of two groups.

Ex: #create a dataset with two groups

```
group1 <- c(2,4,6,8,10)
group2 <- c(1,3,5,7,9)
# perform an F-test to compare group means
f_test <- var.test(group1, group2)
# print the F-statistics and p-value
printf(f_stat)
```

O/P:-

F-test to compare two variances.

data:- group1 and group 2

F = 1, numdf = 4, denomdf = 4, p-value = 1

alternative hypothesis:- true ratio of variances is not equal to 1

95% confidence interval:

0.0041175 9.6045299

sample estimates:-

ratio of variances 1.

9. Explain about correlation matrix.

A. Correlation matrix: Correlation is a table. That displays correlation coefficients between a set of variables. Its range from -1 to 1. -1 indicates strong negative correlati +1 indicates strong positive correlation. 0 indicates a weak or no correlation.

In R, we can use the function for()

:-

eate a dataset with three variables

```
r1 <- c (1,2,3,4,5)
r2 <- c(-2,-4,-6,-8,-10)
r3 <- c(3,6,9,12,15)
 <- data.frame(var1, var2, var3)
rr-matrix <- cor (df)
rint (corr-matrix)
```

O/P:-

	var1	var2	var3
var1	1	-1	1
var2	-1	1	-1
var3	1	-1	1

The cor() function computes the correlation coefficients etween all pairs of variable in the data frame df.

04/08/2023