

NAME: Mugunth.D

REG NO: 40733020

SUBJECT: R PROGRAMMING

DATE : 14-12-2021

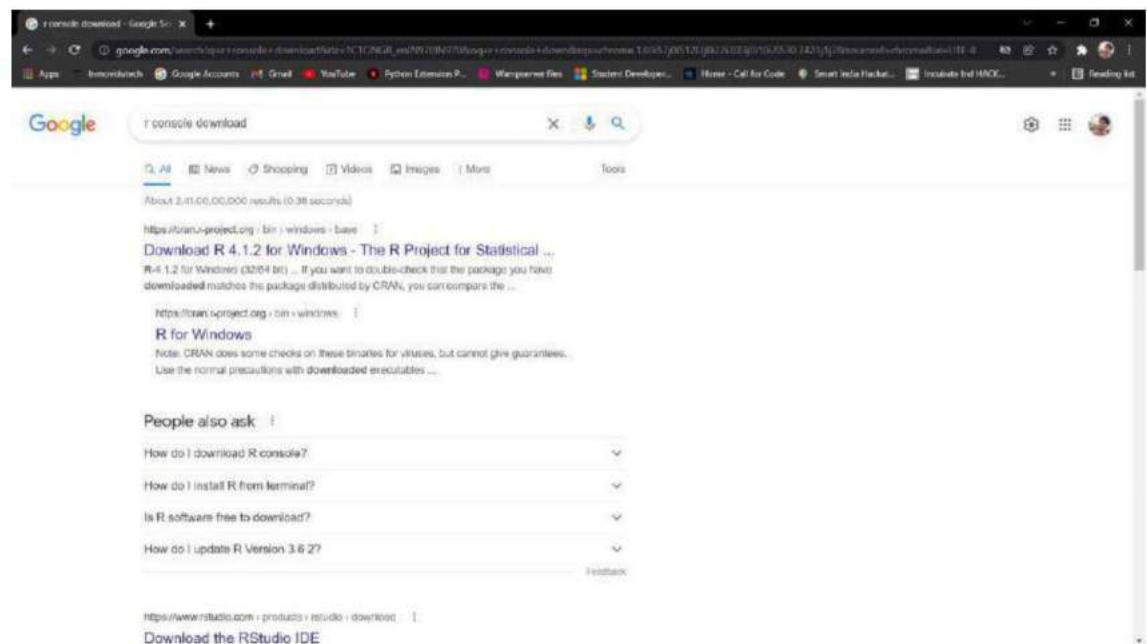
EXPERIMENT NAME: Install R and R Studio and R Basics

Aim:

To install R and R Studio in our system and implement some R Basics in R Studio

INSTALLATION OF R:

STEP 1: SEARCH FOR R CONSOLE IN GOOGLE / BROWSER



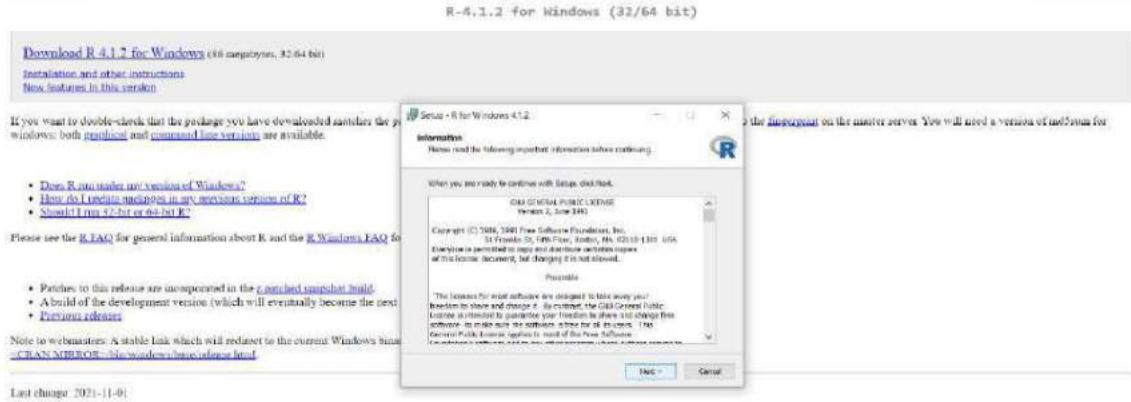
STEP 2: DOWNLOAD R 4.1.2 FOR WINDOWS

The screenshot shows a Microsoft Edge browser window with the URL cran.r-project.org/bin/windows/base/r_4.1.2/. The page title is "Download R 4.1.2 for Windows (32/64 bit)". It features a large download button for "Download R 4.1.2 for Windows (86 megabytes, 32/64 bit)". Below the button are sections for "Installation and other instructions" and "New features in this version". A note at the bottom states: "If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fileserver](#) on the master server. You will need a version of md5sum for windows; both [graphical](#) and [command line](#) versions are available." There is also a "Frequently asked questions" section with links to "Does R run under my version of Windows?", "How do I update packages in my previous version of R?", and "Should I run 32-bit or 64-bit R?". A note for webmasters says: "Note to webmasters: A stable link which will redirect to the current Windows binary release is <[CRAN MIRROR](#)>/bin/windows/base/r_4.1.2.html". At the bottom, it says "Last change: 2021-11-01".

STEP 3: OPEN THE SET-UP DIALOGUE BOX

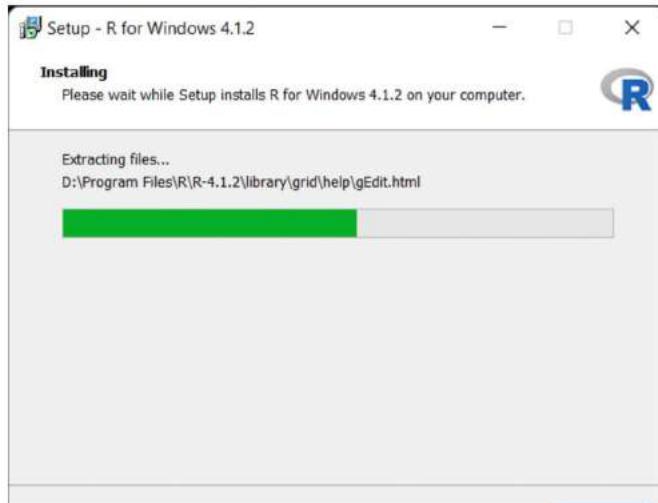
The screenshot shows a standard Windows file download dialog box. The title bar says "R-4.1.2-win.exe". Inside the dialog, there are four buttons: "Open", "Always open files of this type", "Show in folder", and "Cancel". The "Open" button is highlighted with a blue border.

STEP 4: GO THROUGH THE SET-UP PROCESS



Last change: 2021-11-01

STEP 5: WAIT FOR THE INSTALLATION PROCESS



STEP 6: ONCE THE INSTALLATION IS COMPLETED, CLICK ON FINISH



INSTALLATION OF RSTUDIO:

STEP 1: SEARCH FOR RSTUDIO IN GOOGLE / BROWSER

Google search results for "RStudio download":

- [Download the RStudio IDE](https://www.rstudio.com/products/rstudio/download/)
- [RStudio - RStudio](https://www.rstudio.com/products/rstudio/)
- [Download RStudio Desktop Pro](https://www.rstudio.com/download-commercial/)

People also ask:

- Can you download RStudio for free?
- How do I download RStudio?
- Do I need to install R before RStudio?
- Does RStudio work on Mac?

STEP 2: DOWNLOAD RSTUDIO FOR WINDOWS AND OPEN THE SET-UP DIALOGUE BOX

RStudio Desktop 2021.09.1+372 - Release Notes [\[?\]](#)

1. Install R. rstudio requires R 3.0.3 or greater.
2. Download RStudio Desktop. Recommended for your system.

DOWNLOAD RSTUDIO FOR WINDOWS
2021.09.1+372 | 156.9MB

Requires Windows 10 (64-bit)

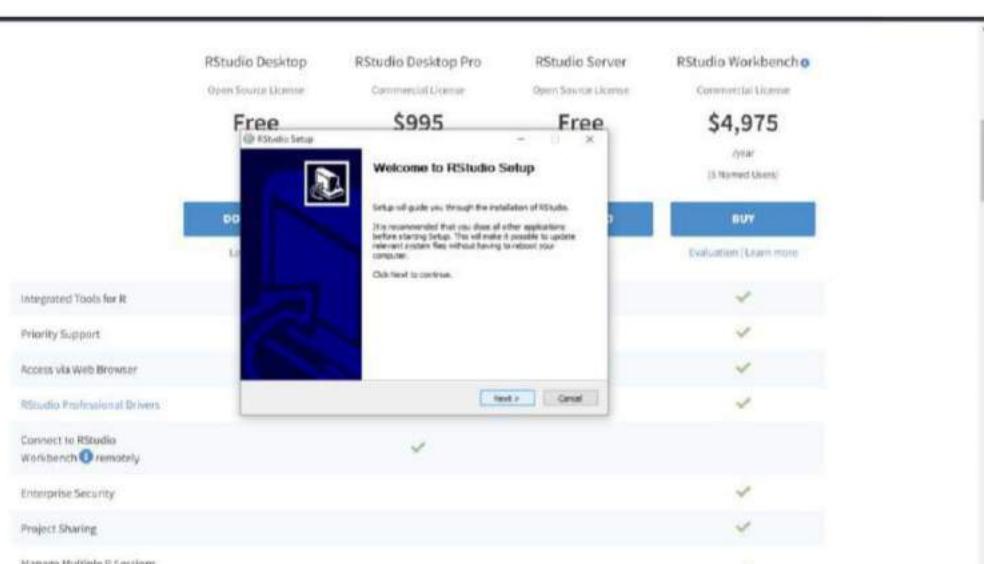
All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

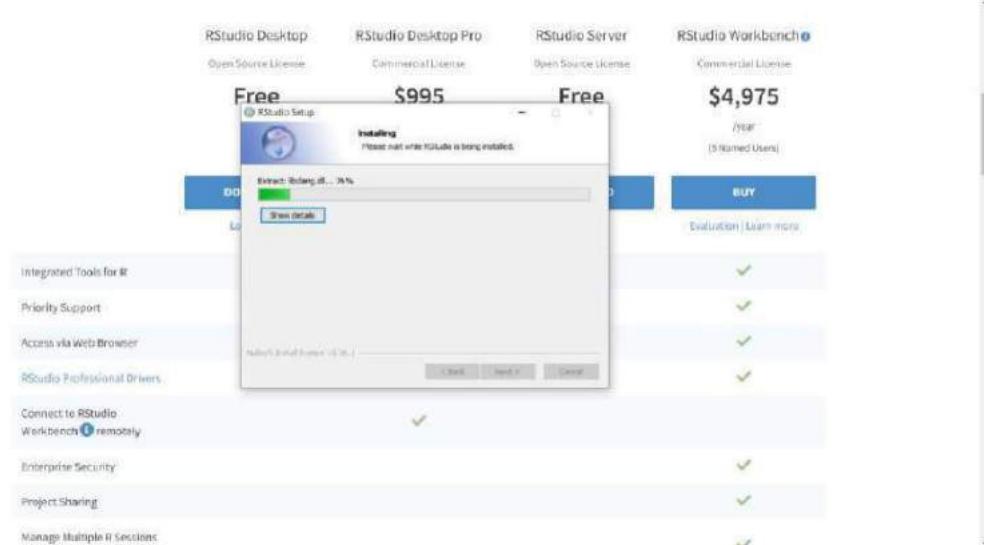
RStudio requires a 64-bit operating system. If you are on a 32-bit system, you can use an older version of RStudio.

OS	Download	Size	SHA-256
Windows 10	RStudio-2021.09.1+372.exe	136.89 MB	1e3d27f5

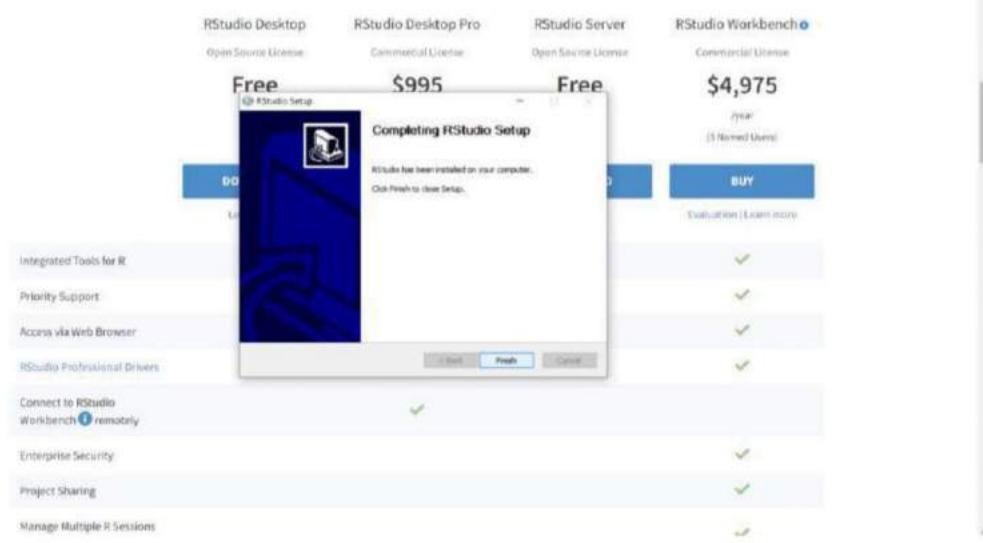
STEP 3: GO THROUGH THE SET-UP PROCESS



STEP 4: WAIT FOR THE INSTALLATION PROCESS

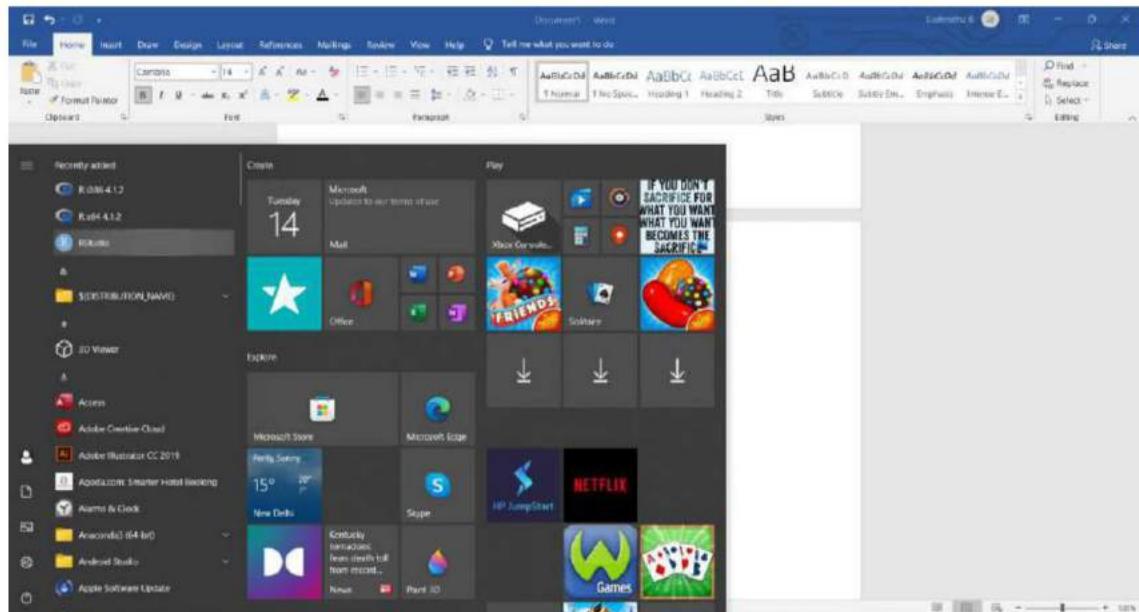


STEP 5: ONCE THE INSTALLATION IS COMPLETED, CLICK ON FINISH

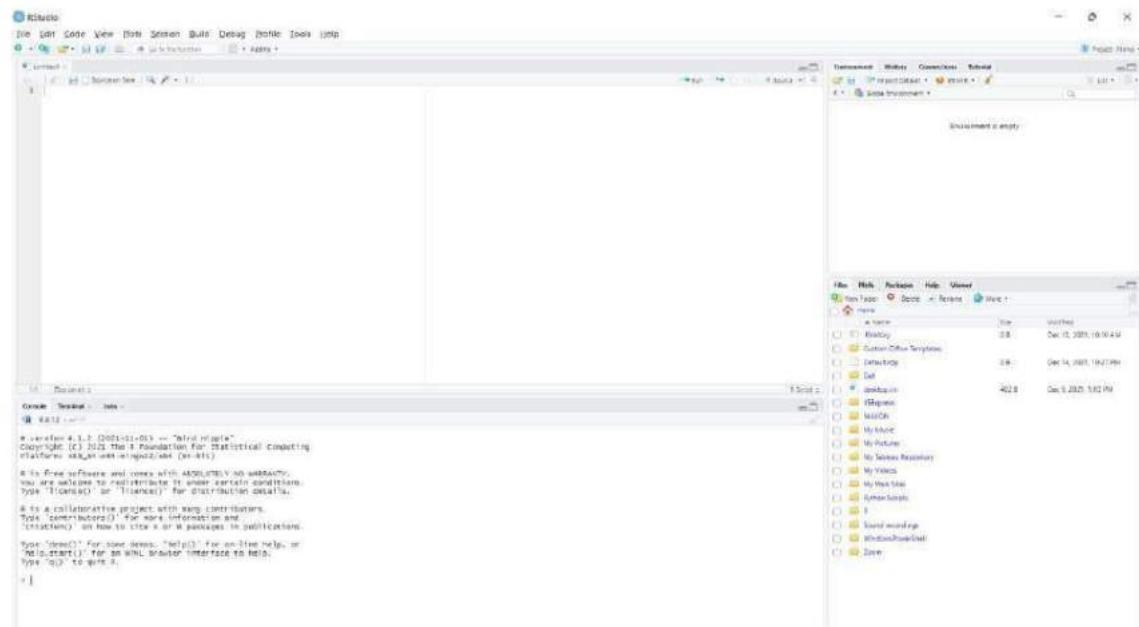


R BASICS IN R STUDIO:

OPEN RSTUDIO

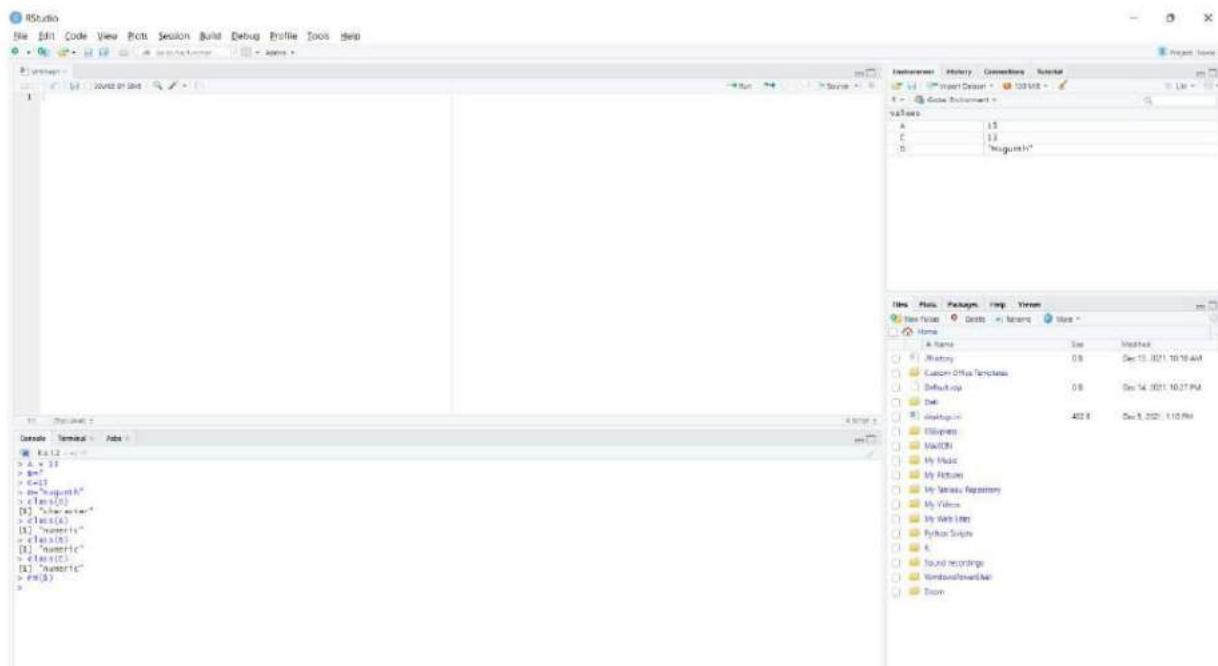


PERFORM YOUR OPERATIONS IN CONSOLE WINDOW



PERFORM OPERATIONS LIKE ASSIGNING A=13, B=7, A<-FALSE, C=13,

D= "Mugunth", class(D), class(A), class(B), class(C), rm(B) IN CONSOLE WINDOW.
THE OUTPUT WILL BE DISPLAYED IN ENVIRONMENT WINDOW



RESULT:

THUS, WE HAVE DOWNLOADED R CONSOLE AND R STUDIO AND IMPLEMENTED THE BASICS OF R STUDIO SUCCESSFULLY.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE : 17-12-2021

EXPERIMENT NAME: R Program using Operators

Aim:

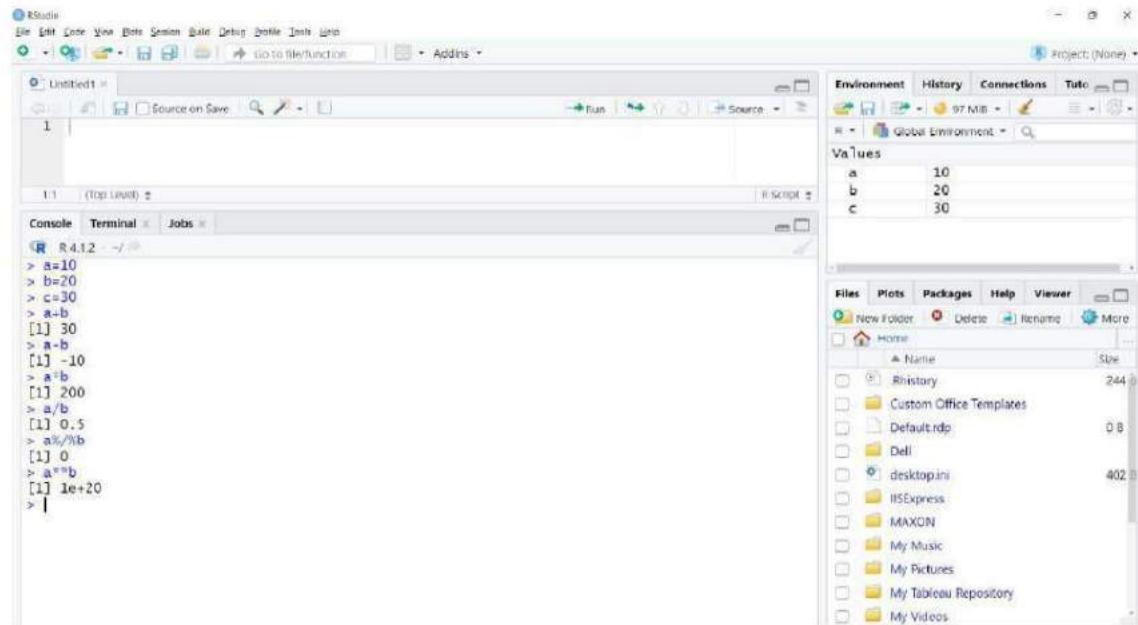
To perform various operations on variables in R language.

Procedure:

Operators are special symbols which represents computations, conditions matching etc. Operators are categorized as Arithmetic, Relational, Logical operators.

1. Arithmetic Operators:

These are mathematical operators that takes two operands and performs calculations. Symbols: +,-,*,/,%,**.



The screenshot shows the RStudio interface with the following details:

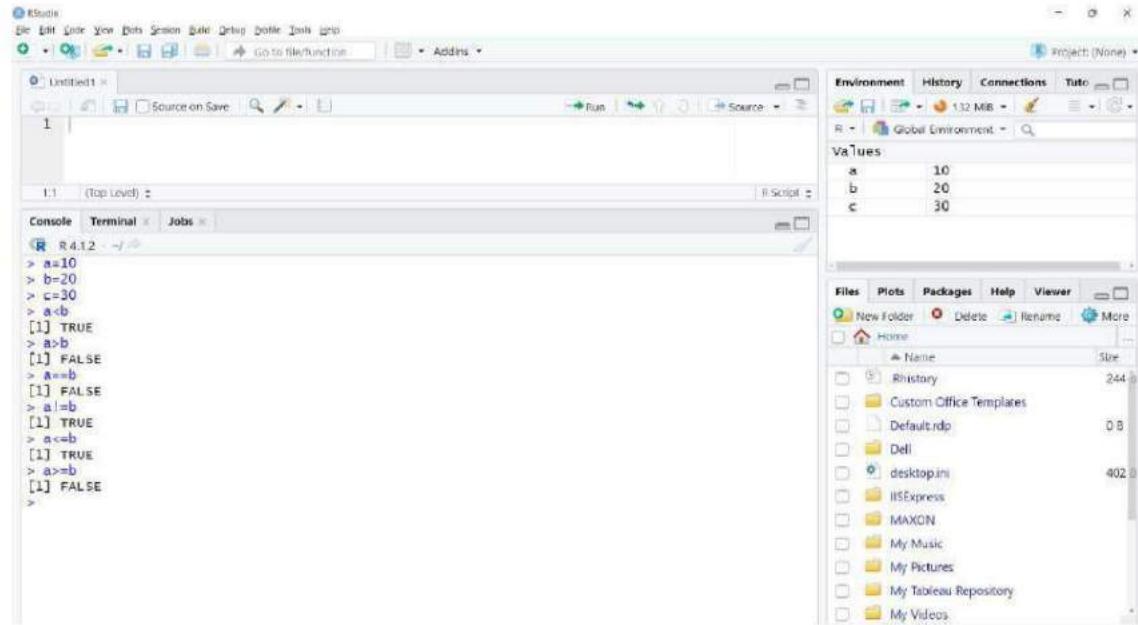
- Console:** Displays R code and its output. The session starts with `R 4.1.2`, followed by:

```
> a=10
> b=20
> c=30
> a+b
[1] 30
> a-b
[1] -10
> a*b
[1] 200
> a/b
[1] 0.5
> a%%b
[1] 0
> a^b
[1] 1e+20
> |
```
- Environment:** Shows variable assignments:

Values	
a	10
b	20
c	30
- File Explorer:** Shows the local file system structure under "Home".

2. Relational Operators:

These are used to check relationship between two operands. If the relation is true it returns true, else returns false. Symbols: <, >, <=, >=, ==, !=.

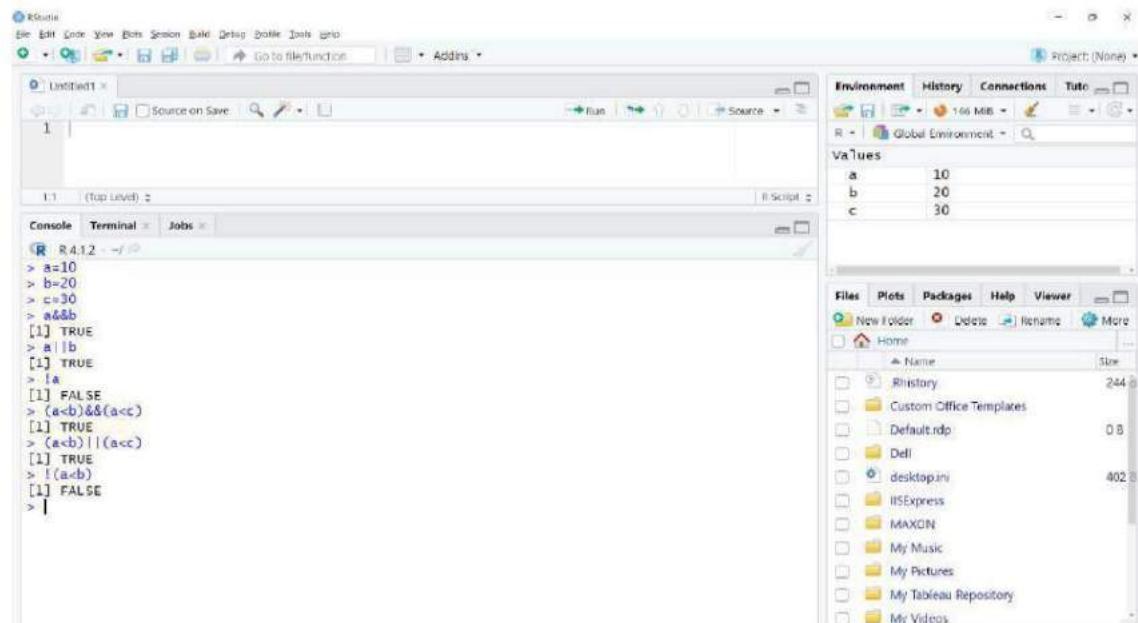


The screenshot shows the RStudio interface. In the top-left, there's a toolbar with various icons. Below it is a menu bar with 'File', 'Edit', 'Code', 'View', 'Bugs', 'Session', 'Build', 'Debug', 'Profile', 'Tools', 'Help'. To the right of the menu is a 'Project' dropdown set to '(None)'. The main area has tabs for 'Console', 'Terminal', and 'Jobs'. The 'Console' tab is active, showing R code and its output. The code includes assignments like 'a<-10', 'b<-20', 'c<-30', and relational checks like 'a<b', 'a==b', etc., which return TRUE or FALSE. To the right of the console is the 'Environment' pane, which lists variables 'a', 'b', and 'c' with their values (10, 20, 30). Below the environment is the 'Files' pane, which shows a file tree with folders like 'History', 'Custom Office Templates', 'Default.rdp', 'Dell', 'desktop.ini', 'IISExpress', 'MAXON', 'My Music', 'My Pictures', 'My Tableau Repository', and 'My Videos'. The bottom of the interface has a status bar.

```
> a<-10
> b<-20
> c<-30
> a<b
[1] TRUE
> a>b
[1] FALSE
> a==b
[1] FALSE
> a!=b
[1] TRUE
> a<=b
[1] TRUE
> a>=b
[1] FALSE
>
```

3. Logical Operators:

These operators performs operations on the given expressions. There are 3 logical operators: and, or, not.



This screenshot is identical to the one above, showing the RStudio interface with the 'Console' tab active. The code entered and its output remain the same, demonstrating the use of logical operators like '&&' (and), '| |' (or), and '!' (not) on the variables 'a', 'b', and 'c'.

```
> a<-10
> b<-20
> c<-30
> a&&b
[1] TRUE
> a | | b
[1] TRUE
> !a
[1] FALSE
> (a<b)&&(a<c)
[1] TRUE
> (a>b)| |(a>c)
[1] TRUE
> !(a>b)
[1] FALSE
> |
```

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

REG NO: 40733020

SUBJECT: R PROGRAMMING

DATE: 21-12-2021

EXPERIMENT NAME: R Program for Vectors and Matrices Operations.

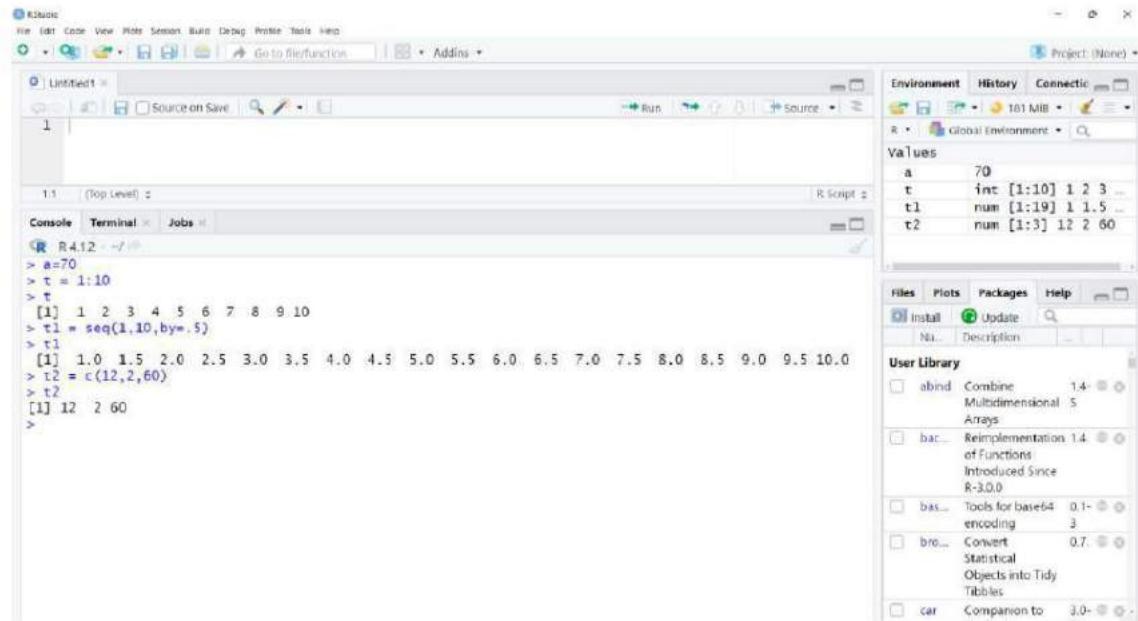
AIM: To perform Vectors and Matrices Operations in R programming.

Procedure:

- **To perform Vector Operations.**
- **To perform Matrices Operations.**

Vectors:

- **Vectors can be created using operators such as :, seq(), c().**



The screenshot shows the RStudio interface. In the top-left, there's a menu bar with File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help. Below it is a toolbar with various icons. The main area has tabs for 'Console' (selected), 'Terminal', and 'Jobs'. The 'Console' tab shows R code and its output:

```
R 4.12 - ~/r/R-4.1.2/bin/R
> a=70
> t = 1:10
> t
[1] 1 2 3 4 5 6 7 8 9 10
> t1 = seq(1,10,by=.5)
> t1
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
> t2 = c(12,2,60)
> t2
[1] 12  2 60
```

On the right side, there's an 'Environment' pane showing variables:

Values
a 70
t int [1:10] 1 2 3 ...
t1 num [1:19] 1 1.5 ...
t2 num [1:3] 12 2 60

Below the environment pane, there are tabs for 'Files', 'Plots', 'Packages', and 'Help'. Under 'User Library', several packages are listed with their versions and descriptions.

- Accessing Elements from a vector: Elements of the vector can be accessed using their index values.

The screenshot shows the RStudio interface with the following R session history:

```

1
1.1 (Top Level) >
> t
[1] 1 2 3 4 5 6 7 8 9 10
> t[6]
[1] 6
> p = c(10,20,30,40,50)
> p[2]
[1] 20
> p[c(2,5)]
[1] 20 50
> p[1:4]
[1] 10 20 30 40
> p[c(-2,-5)]
[1] 10 30 40
> t = p[c(-2)]
> t
[1] 10 30 40 50
> p[-1]
[1] 20 30 40 50
> 

```

The Environment pane shows variables: a (70), p (num [1:5] 10 20 30 40 50), t (num [1:4] 10 30 40 50), t1 (num [1:19] 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10), and t2 (num [1:3] 12 2 60).

- Vectors Operations: We can perform arithmetic, relational, logical operations between vectors.

The screenshot shows the RStudio interface with the following R session history:

```

1
1.1 (Top Level) >
> g<-10
[1] 62 71 88 36 25 49
> g<-4
[1] 47 56 73 21 10 34
> g<-t
[1] 104 122 156 52 30 78
> g<-r
[1] 17.333333 20.333333 26.000000 8.666667 1.000000 13.000000
> g<-q
[1] 7312616 13845841 37015056 456976 50623 2313441
> s <-c(1,2,3,4,5,6)
> t<-s
> g<-s
[1] 53 63 81 30 20 45
> g<-s
[1] 51 59 75 22 10 33
> g<-t
[1] 52 123 234 104 75 234
> g</-
[1] 52.0 30.5 26.0 6.5 3.0 6.5
> g</s
[1] 52 3721 474552 456976 759375 3518743761
> g<%>%
[1,]
[1,] 821
> 

```

The Environment pane shows variables: a (70), g (num [1:6] 52 63 81 78 20 45), p (num [1:5] 10 20 30 40 50), s (num [1:6] 1 2 3 4 5 6), t (num [1:4] 10 30 40 50), t1 (num [1:19] 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10), t2 (num [1:3] 12 2 60), and r (num [1:19] 17.333333 20.333333 26.000000 8.666667 1.000000 13.000000).

Matrix:

- **Creating Matrix:**

Syntax: `matrix(data, nrow, ncol, byrow, dimnames)`.

- Operations on matrix:

Basic arithmetic operations such as addition, subtraction, multiplication, division can be performed between matrices. We can also perform transpose of a matrix and inverse of a matrix.

The figure shows a screenshot of a computer desktop with two windows open. On the left is a web browser displaying a Shiny application titled 'Jobs'. The application has a header 'Jobs' and a table with columns 'Job ID', 'Title', 'Category', and 'Status'. A 'Run' button is at the bottom. On the right is the RStudio IDE. The top menu bar includes 'File', 'Edit', 'Code', 'View', 'Plots', 'Session', 'Build', 'Group', 'Tools', 'Tools...', 'Tools...', and 'Help'. The main workspace shows the R code for the 'Jobs' app, which includes functions for generating random data and creating a Shiny UI. Below the code is a 'Data' viewer showing a sample dataset with columns 'ID', 'Title', 'Category', and 'Status'. At the bottom of the RStudio interface is a file browser with several files listed.

Result:

Thus, we have performed vector and matrix operations in R successfully.

NAME: Mugunth. D
REG NO: 40733020

SUBJECT: R PROGRAMMING
DATE: 29-12-2021

EXPERIMENT NAME: R Program for List and String Operations.

Aim:

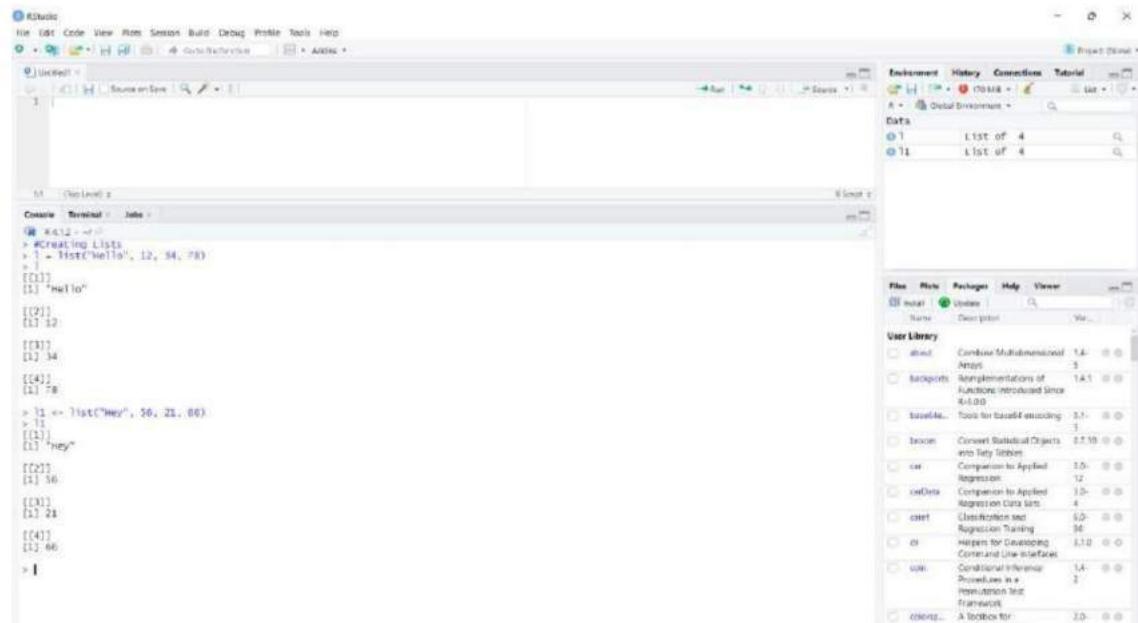
To perform various operations on List and String .

Procedure:

- To perform List Operations.**
- To perform String Operations.**

List Operations:

- Creating a List**



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Run, Session, Build, Debug, Profile, Tools, Help, and a Project pane. The left sidebar has tabs for Home, Code, Data, and Help. The main workspace shows the following R code and its output:

```
RStudio
File Edit Code View Run Session Build Debug Profile Tools Help
Home Code Data Help
Project Home
Source on Dev 1234567890
1
[[1]] "Hello"
[[2]]
[[3]] 34
[[4]] 78
> t1 <- list("Hey", 10, 21, 66)
> t1
[[1]] "hey"
[[2]]
[[3]] 34
[[4]] 78
[[5]] 21
[[6]] 66
> |
```

The right pane displays the Environment tab, showing two objects: t1 (LIST OF 4) and t2 (LIST OF 4). The User Library pane lists several packages with their versions and descriptions:

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-5
backports	Reimplementations of Functions Introduced Since R 4.0.0	1.4.1
base64enc	Tools for base64 Encoding	2.1.1
bitacode	Convert Statistical Objects into Text Tables	2.7.10
car	Computation for Applied Regression	2.5-
carData	Computation for Applied Regression Data Sets	3.0-
caret	Classification and Regression Training	4.0-
cli	Command Line Interface	2.1.0
combn	Conditional Inference Procedures in a Permutation Test Framework	1.4-
colorspace	A Toolbox for Computing with Colourspace	2.0-

RStudio interface showing the Environment pane with a list of three items:

- 1 List of 4
- 11 List of 4
- 12 List of 3

The Console pane shows the following R code and output:

```
R> l2 = list(c(1,2,3), matrix(c(1,2,3,4), nrow = 2), list("Hello", 45))
R> l2
[[1]]
[1] 1 2 3

[[2]]
[,1] [,2]
[1,]    1    3
[2,]    2    4

[[3]]
[[3]][[1]]
[1] "Hello"

[[3]][[2]]
[1] 45

> |
```

• Accessing Elements of a List

RStudio interface showing the Environment pane with a list of three items:

- 1 List of 4
- 11 List of 4
- 12 List of 3

The Console pane shows the following R code and output, demonstrating how to access elements of a list:

```
R> #Accessing Elements of a List
R> l2[[1]]
[[1]]
[1] 1 2 3

R> l2[[1]][[2]]
[1] 2
R> l2[[1]][[3]]
[1] 3
R> l2[[2]][[2]][[3]]
[1] 3
R> l2[[3]][[2]]
[[1]]
[1] 45

R> l2[[3]][[1]]
[[1]]
[1] "Hello"

R> l2[[2]][[4]]
[1] 4
> |
```

• Updating Elements of a List

The screenshot shows the RStudio interface with the following session history:

```
R 4.1.2 -->
> L[1] = 54
> L[4] = "Hellooo"
> L
[[1]]
[1] "Hello"

[[2]]
[1] 12

[[3]]
[1] 34

[[4]]
[1] 78

> L[2] = "Hi"
> L
[[1]]
[1] "Hello"

[[2]]
[1] "Hi"

[[3]]
[1] 34

[[4]]
[1] 78
```

The Data pane shows three lists:

- L1: List of 4
- L1: List of 4
- L2: List of 3

The screenshot shows the RStudio interface with the following session history:

```
R 4.1.2 -->
> L[1] = 54
> L[4] = "Hellooo"
> L
[[1]]
[1] "Hello"

[[2]]
[1] 12

[[3]]
[1] 34

[[4]]
[1] "Hellooo"

> |
```

The Data pane shows three lists:

- L1: List of 4
- L1: List of 4
- L2: List of 3

- **Deleting Elements of a List**

The screenshot shows the RStudio interface with the following code in the console:

```
R> #Deleting Elements of a List
> 
> [[1]]
[[1]] "Hello"
[[2]]
[[1]] "Hi"
[[3]]
[[1]] 54
[[4]]
[[1]] "Hellooo"
> [[2]] = NULL
> 
> [[1]]
[[1]] "Hello"
[[2]]
[[1]] 54
[[3]]
[[1]] "Hellooo"
>
```

The Data pane on the right shows three lists named '1', '11', and '12'.

String Operations:

- **Creating a String**

The screenshot shows the RStudio interface with the following code in the console:

```
R> #Creating a String
> s = "Hubble"
> s1 <- "Bubble"
> 
```

The Data pane on the right shows two lists named '1' and '11'.

- **Operations:**

- **length(x):** Get or set the length of vectors (including lists) and factors, and of any other R object for which a method has been defined.
- **nchar(x):** returns number of characters in a string.
- **toupper(x):** translates all the characters of a string to upper case.
- **tolower(x):** translates all the characters of a string to lower case.
- **chartr(x):** translates each character in x that is specified in old to the corresponding character specified in new. Ranges are supported in the specifications.
- **substr(x):** Extract or replace substrings in a character vector.
- **Concatenating Strings:** two or more strings can be concatenated using paste(x) command.
- **strsplit(x):** Split the elements of a character vector x into substrings according to the matches to substring split within them.

The screenshot shows the RStudio interface with the following session history:

```
R 4.1.2 - /~/
> s2 = c("Hubble", "James Webb")
> length(s2)
[1] 2
> nchar(s2)
[1] 6 10
> toupper("Esa")
[1] "ESA"
> tolower("Nasa")
[1] "nasa"
> chartr("H", "B", "Hubble")
[1] "Bubble"
> substr("Hubble", 1, 3)
[1] "Hub"
> paste("Hey", "Hubble")
[1] "Hey Hubble"
> strsplit("Hey dude wassup", " ")
[[1]]
[1] "Hey"     "dude"    "wassup"
```

The right pane of RStudio displays the Global Environment, showing objects 1, 11, and 12, all of which are lists. It also shows the User Library with several packages listed.

Result:

Thus we have performed operations on List and Strings successfully.

NAME: Mugunth. D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 29-12-2021

EXPERIMENT NAME: R Program for Data Frame Operations.

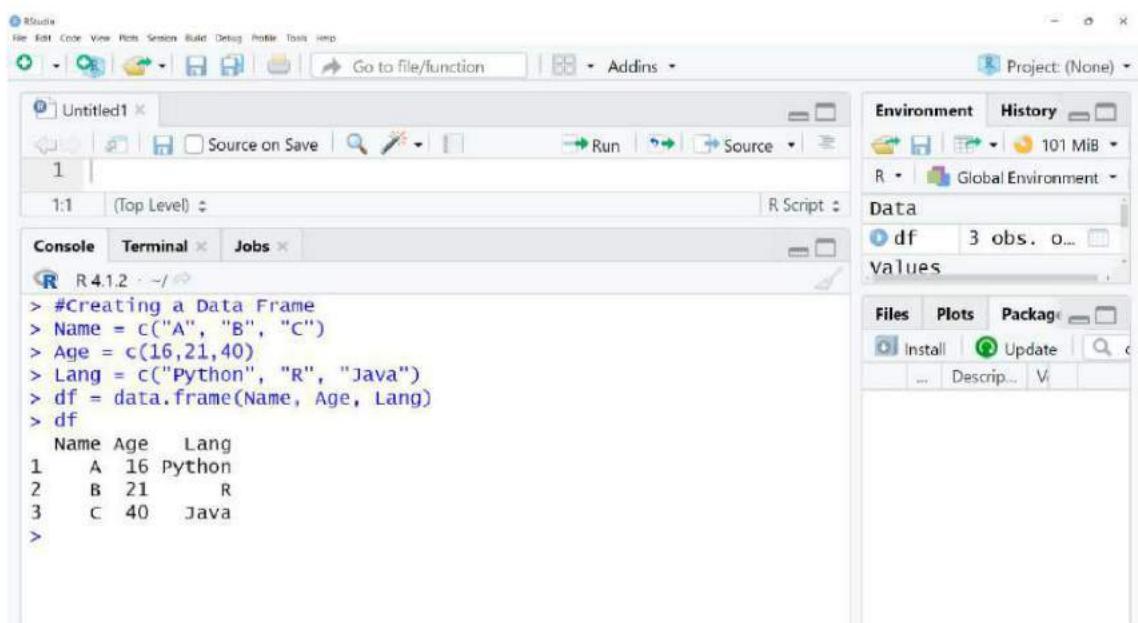
Aim:

To perform various operations using Data Frames.

Procedure:

- The function `data.frame()` creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.
- Syntax:
`data.frames(..., row.names = NULL, check.rows = FALSE,
check.names = TRUE, fix.empty.names = TRUE,
stringsAsFactors = FALSE)`

➤ Creating a Data Frame



The screenshot shows the RStudio interface with the following details:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, New, and Run.
- Console Tab:** Shows the R session output:

```
R 4.1.2 -->
> #Creating a Data Frame
> Name = c("A", "B", "C")
> Age = c(16,21,40)
> Lang = c("Python", "R", "Java")
> df = data.frame(Name, Age, Lang)
> df
```

	Name	Age	Lang
1	A	16	Python
2	B	21	R
3	C	40	Java
- Environment Tab:** Shows the global environment with a data frame named `df`.
- Plots Tab:** Shows a small preview of a scatter plot.

➤ Adding/Combining Elements by rows/columns

The screenshot shows the RStudio interface with the following details:

- File menu:** File, Edit, Code, View, Data, Session, Date, Debug, Profile, Tools, Help.
- Toolbar:** Undo, Redo, Cut, Copy, Paste, Find, Go to file/function, Addins.
- Console:** Displays R code and its output. The code demonstrates concatenating data frames (rbind) and combining data frames by columns (cbind).

```
R> #Adding/Combining Elements by rows
> df1 = rbind(df, data.frame(Name = "D", Age = 25, Lang = "C++"))
> df1
  Name Age Lang
1    A  16 Python
2    B  21      R
3    C  40 Java
4    D  25 C++
> #Adding/Combining Elements by columns
> df2 = cbind(df1, City = c("chennai", "Mumbai", "Delhi", "Calcutta"))
> df2
  Name Age Lang     City
1    A  16 Python Chennai
2    B  21      R Mumbai
3    C  40 Java   Delhi
4    D  25 C++ Calcutta
```
- Environment pane:** Shows the Global Environment with objects df, df1, and df2.
- Data pane:** Shows the contents of df, df1, and df2.
- Plots pane:** Empty.
- Packages pane:** Empty.

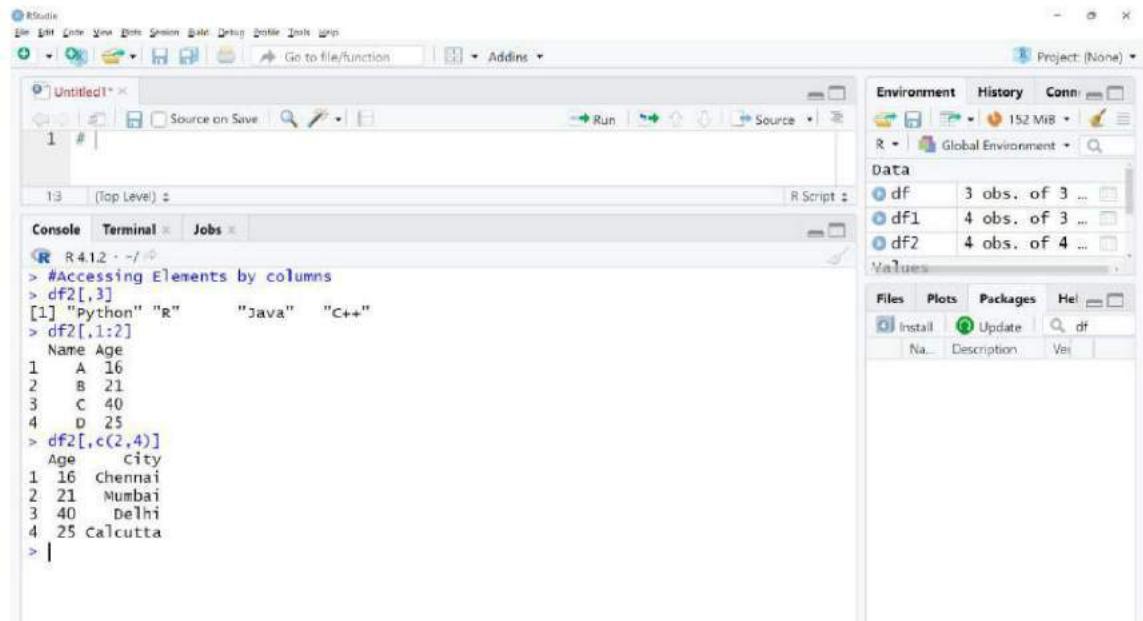
➤ Accessing Elements by rows

The screenshot shows the RStudio interface with the following details:

- File menu:** File, Edit, Code, View, Data, Session, Date, Debug, Profile, Tools, Help.
- Toolbar:** Undo, Redo, Cut, Copy, Paste, Find, Go to file/function, Addins.
- Console:** Displays R code and its output. The code demonstrates how to access specific rows of a data frame.

```
R> #Accessing Elements by rows
> df2[1,]
  Name Age Lang     city
1    A  16 Python Chennai
> df2[1:3,]
  Name Age Lang     city
1    A  16 Python Chennai
2    B  21      R Mumbai
3    C  40 Java   Delhi
> df2[c(1,3),]
  Name Age Lang     city
1    A  16 Python Chennai
3    C  40 Java   Delhi
```
- Environment pane:** Shows the Global Environment with objects df, df1, and df2.
- Data pane:** Shows the contents of df, df1, and df2.
- Plots pane:** Empty.
- Packages pane:** Empty.

➤ Accessing Elements by columns



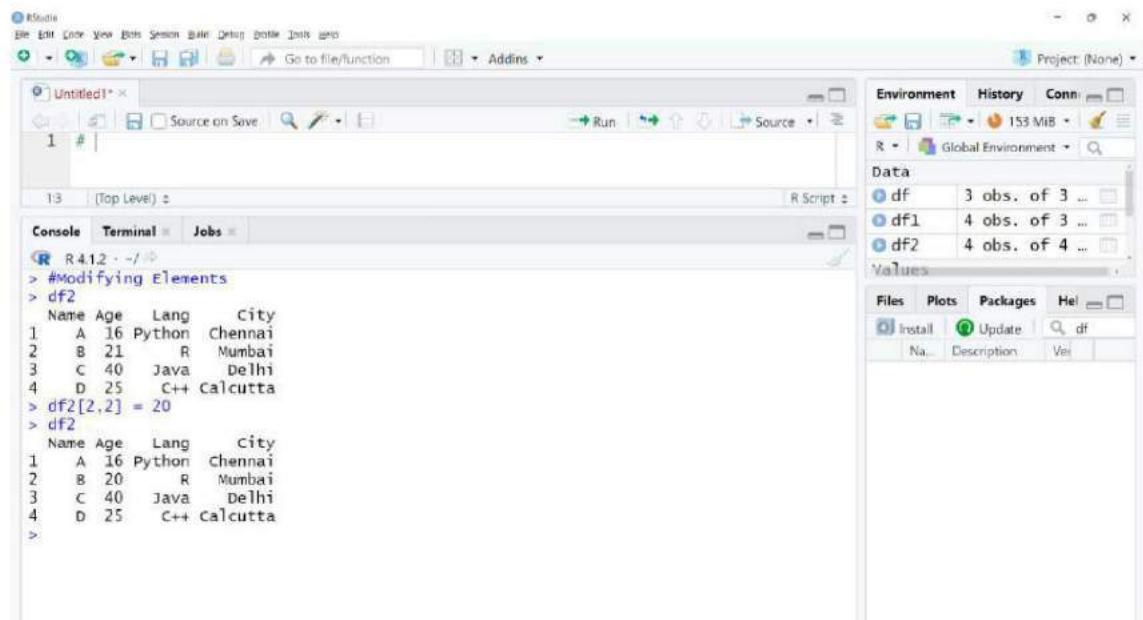
The screenshot shows the RStudio interface with the following details:

- Console:** Displays R code and its output. The code demonstrates how to access elements by columns in a data frame. It includes:
 - Creating a data frame `df2` with columns `Name`, `Age`, `Lang`, and `City`.
 - Accessing the first three columns with `df2[,1:3]`, resulting in a vector: [1] "python" "R" "java" "C++".
 - Accessing the first two columns with `df2[,1:2]`, resulting in a data frame:

Name	Age
A	16
B	21
C	40
D	25
 - Accessing columns 2 and 4 with `df2[,c(2,4)]`, resulting in a data frame:

Age	City
16	Chennai
21	Mumbai
40	Delhi
25	Calcutta
- Data View:** Shows the data frames `df`, `df1`, and `df2` in the Global Environment.

➤ Modifying Elements of a Data Frames



The screenshot shows the RStudio interface with the following details:

- Console:** Displays R code and its output. The code demonstrates how to modify elements in a data frame. It includes:
 - Creating a data frame `df2` with columns `Name`, `Age`, `Lang`, and `City`.
 - Modifying the value at index [2,2] with `df2[2,2] = 20`, resulting in a data frame:

Name	Age	Lang	City
A	16	Python	Chennai
B	21	R	Mumbai
C	40	Java	Delhi
D	25	C++	Calcutta
 - Modifying the value at index [2,2] again with `df2[2,2] = 20`, resulting in a data frame:

Name	Age	Lang	City
A	16	Python	Chennai
B	20	R	Mumbai
C	40	Java	Delhi
D	25	C++	Calcutta
- Data View:** Shows the data frames `df`, `df1`, and `df2` in the Global Environment.

➤ Removing a row

The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1.R' script with the following code:

```
R> #Removing a row
R> df3 = df2[-4,]
R> df3
  Name Age Lang   City
1   A   16 Python Chennai
2   B   20      R Mumbai
3   C   40 Java   Delhi
```

In the top-right pane, the 'Environment' tab shows four data frames: df (3 obs. of 3), df1 (4 obs. of 3), df2 (4 obs. of 4), and df3 (3 obs. of 4). The 'Files' tab shows a file named 'df'.

➤ Removing a column

The screenshot shows the RStudio interface. In the top-left pane, there is an 'Untitled1.R' script with the following code:

```
R> #Removing a column
R> df4 = df2[,-2]
R> df4
  Name Lang   City
1   A Python Chennai
2   B      R Mumbai
3   C Java   Delhi
4   D C++ Calcutta
```

In the top-right pane, the 'Environment' tab shows five data frames: df (3 obs. of 3), df1 (4 obs. of 3), df2 (4 obs. of 4), df3 (3 obs. of 4), and df4 (4 obs. of 3). The 'Files' tab shows a file named 'df'.

Result:

Thus we have performed operations on Data Frames successfully.

NAME: Mugunth.D

REG NO: 40733020

SUBJECT: R PROGRAMMING

DATE: 04-01-2021

EXPERIMENT NAME: R Program for Built-in functions and User defined functions.

Aim:

To

- a) Write the R Script for built in functions (Any 10)**
- b) Write a R program to check odd or even number using custom Function (User Defined Function)**

Procedure:

a) Built-in Functions:

There are various built-in functions available in R. Here is a few popular built-in functions used in R:

- max(x) – returns the maximum of the input values.**
- min(x) – returns the minimum of the input values.**
- sum(x) – returns the sum of the input values.**
- sin(x) – returns the sin of input values.**
- cos(x) – returns the cosine of input values.**
- tan(x) – returns the tangent of input values**
- floor(x) – returns the largest integer less than or equal to input values.**
- ceiling(x) – returns the smallest number greater than or equal to input values.**
- round(x) - rounds the values in its first argument to the specified number of decimal places.**
- sqrt(x) – returns the square root of input values which is greater than zero (0).**

RStudio interface showing a console window with the following R script and output:

```
R 4.1.2 -->
> # write the R Script for built in functions(Any 10)
> max(23,54,71,36,32)
[1] 71
> min(-28,33,-32,-17)
[1] -28
> sum(12,28,73,81)
[1] 194
> sin(pi/3)
[1] 0.8660254
> cos(pi)
[1] -1
> tan(pi/3)
[1] 1.732051
> round(-18.5)
[1] -19
> floor(-23.2)
[1] -23
> ceiling(-26.7)
[1] -26
> sqrt(36)
[1] 6
> |
```

The right side of the interface shows the "Environment" tab with "Environment is empty". Below it is the "Packages" tab, which lists various R packages with their versions and status.

b) Write a R program to check odd or even number using custom Function (User Defined Function)

RStudio interface showing a console window with the following R script and output:

```
1 # Write a R program to check odd or even number using custom Function(User Defined Function).
2
3 oddeven = function(n){
4   if( n%%2 == 0){
5     print("the number is Even")
6   }else{
7     print("the number is odd")
8   }
9 }
10 a = as.integer(readline(prompt = "Enter a number:"))
11 oddeven(a)
```

The right side of the interface shows the "Environment" tab with variable "a" set to 36L and the function "oddeven". Below it is the "Packages" tab, which lists various R packages with their versions and status.

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

REG NO: 40733020

SUBJECT: R PROGRAMMING

DATE: 04-01-2021

EXPERIMENT NAME: R Program for Control Structures.

Aim:

To

- a) Write the R Script to check for voting Eligibility.**
- b) Write the R Script to find Youngest among given three children.**
- c) Write the R script to sort values of given vector. (Without using built in function)**

Procedure:

a) R Script to check for voting Eligibility

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plot, Session, Build, Debug, Profile, Tools, Help, and a Help icon. The main area has tabs for 'Untitled.R' (selected), 'Source on Save', and 'Run'. Below the tabs is a search bar. The code editor contains the following R script:

```
1 # Write the R Script to check for voting Eligibility
2
3 age <- as.integer(readline(prompt = "Enter your age :"))
4
5 if (age >= 18) {
6   print(paste("You are valid for voting :", age))
7 } else {
8   print(paste("You are not valid for voting :", age))
9 }
```

The R console at the bottom shows the execution of the script and its output:

```
R412> source("./active-rstudio-document")
Enter your age :19
[1] "you are valid for voting : 19"
> source("./active-rstudio-document")
Enter your age :12
[1] "you are not valid for voting : 12"
> |
```

On the right side, there are several panes: Environment, History, Connections, Tutorial, and a Global Environment pane showing 'age' with value '12L'. Below the environment pane is a package browser window titled 'Packages' with tabs for 'File', 'Plots', 'Packages', 'Help', and 'Viewer'. It lists packages such as abind, assertthat, base64enc, bit, bit64, blob, broom, and callr from R, along with their descriptions and versions.

b) R Script to find Youngest among given three children

The screenshot shows the RStudio interface with the following details:

- Code Editor:** An R script titled "Untitled" is open. The code prompts the user for ages of three children and prints the youngest. It uses `readline` for input and `if` statements for comparison.
- Environment View:** Shows variables `a`, `age`, `b`, `c`, and `d` with their respective values: "12", "12L", "16", "15", and a character vector `chr [1:3] "12" "13" "15".
- Console:** Displays the execution of the script and its output, showing the youngest child for different inputs.
- Packages View:** Shows a list of installed packages, including abind, digest, assertthat, backports, base64enc, bit, bit64, blob, broom, and callr.

c) R script to sort values of given vector (Without using built in function)

The screenshot shows the RStudio interface with the following details:

- Code Editor:** An R script titled "Untitled" is open. The code defines a function `bs` that sorts a vector `x` using a bubble sort algorithm.
- Environment View:** Shows variables `a` (vector [1:5] 90 68 72 51 28), `s` (vector [1:5] 28 51 68 72 90), and the function `bs`.
- Console:** Displays the execution of the script and its output, showing the sorted vector.
- Packages View:** Shows a list of installed packages, including digest, assertthat, backports, base64enc, bit, bit64, blob, broom, and callr.

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 08-01-2022

EXPERIMENT NAME: R Program for Control Structures Part-II

Aim:

To

- a) Write the R Script to search an element using Linear Search (Use for, if and break)**
- b) Write the R Script to search an element using Binary Search (use while, if and break)**

Algorithm:

a) Linear Search:

- Step 1: Set i to 1**
- Step 2: if i > n then go to step 7**
- Step 3: if A[i] = x then go to step 6**
- Step 4: Set i to i + 1**
- Step 5: Go to Step 2**
- Step 6: Print Element x Found at index i and go to step 8**
- Step 7: Print element not found**
- Step 8: Exit.**

b) Binary Search:

- Step 1: Create a Function rs with arguments lst, value**
- Step 2: Set low=1**
- Step 3: Set high=length(lst)**
- Step 4: Set mid=length(lst)%/2**
- Step 5: if (lst[low]==value) then return low**
- Step 6: if (lst[high]==value) then return high**
- Step 7: else while (lst[mid] != value) and**
- Step 7.1: if (value > lst[mid]) return low = mid+1**
- Step 7.2: if (value < lst[mid]) return high = mid - 1**
- Step 7.3: if(high<low) return mid=-1;break**
- Step 7.4: mid=(low+high)%/2**

Step 7: Return mid.

Procedure:

a) R Script to search an element using Linear Search

The screenshot shows the RStudio interface with the following details:

- Code View:** The code editor displays a script named "Untitled.R" containing a linear search function. The function takes a vector `v` and a value `x` as input. It iterates through the vector using a for loop, printing each element and checking if it matches `x`. If a match is found, it prints the index and returns. If no match is found after the loop, it prints a message indicating the value was not found.
- Environment View:** Shows the global environment with variables `v` and `x` defined. `v` is a numeric vector [1:6] containing 2, 3, 5, 7, 9, 1. `x` is a numeric value 7.
- Console View:** The console shows the execution of the script. It prompts for the element to be searched and then lists the elements of the vector `v`.
- File Library View:** Shows the user library with packages like `grid`, `gridExtra`, `assertthat`, `backports`, and `bit64` installed.

b) R Script to search an element using Binary Search

The screenshot shows the RStudio interface with the following details:

- Code View:** The code editor displays a script named "Untitled.R" containing a binary search function. The function takes a vector `v` and a value `x` as input. It initializes `low` to 1 and `high` to the length of the vector. It then enters a while loop where it calculates the middle index `mid` and compares the value at `v[mid]` with `x`. If they are equal, it prints the index and exits. If `v[mid]` is less than `x`, it sets `low` to `mid+1`. If `v[mid]` is greater than `x`, it sets `high` to `mid-1`. The loop continues until `low` is greater than `high`.
- Environment View:** Shows the global environment with variables `v` and `x` defined. `v` is a numeric vector [1:6] containing 2, 3, 5, 7, 9, 1. `x` is a numeric value 7.
- Console View:** The console shows the execution of the script. It prompts for the values to be searched and then lists the elements of the vector `v`.
- File Library View:** Shows the user library with packages like `grid`, `gridExtra`, `assertthat`, `backports`, and `bit64` installed.

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 11-01-2022

EXPERIMENT NAME: R Script for Data Visualization

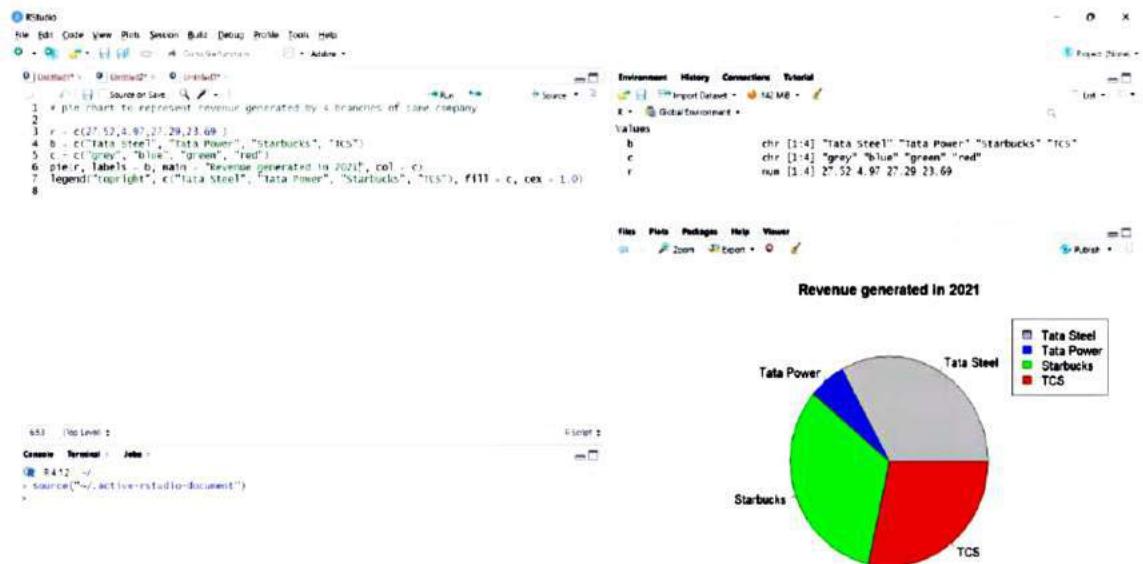
Aim:

To

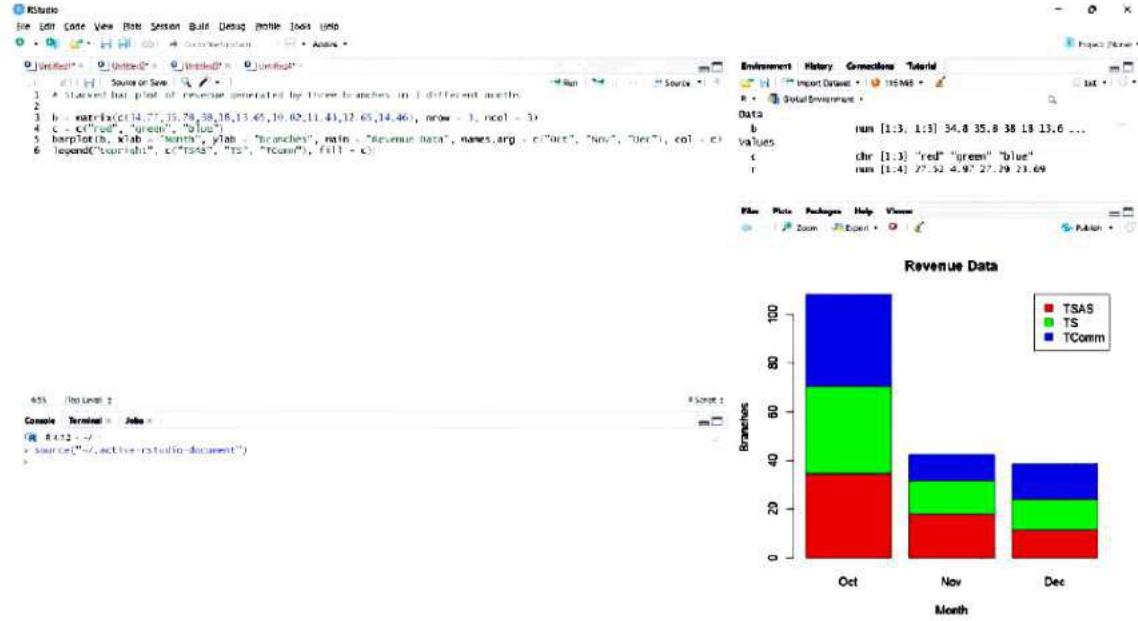
- Write the R Script to generate pie chart to represent revenue generated by 4 branches of same company. Provide appropriate legend also.
- Write the R Script to generate Stacked bar plot to represent revenue generated by three branches in 3 different months.
- Write the R script to show mark distribution of students in 3 different sections using line graph.

Procedure:

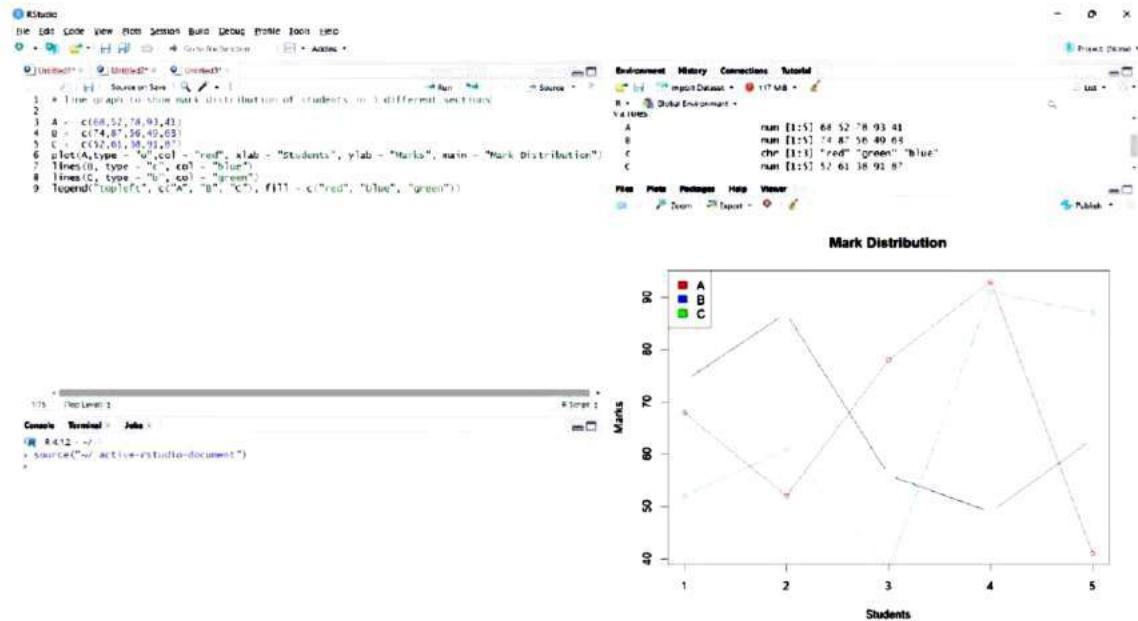
- R Script to generate pie chart of revenue generated by 4 branches of same company



b) R Script to generate Stacked bar plot of revenue generated by three branches in 3 different months



c) R script to show mark distribution of students in 3 different sections using line graph



Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 21-01-2021

EXPERIMENT NAME: R Script for Data Visualization-Par-II

Aim:

To

- a) Write the R Script to generate Scatter Plot to show relationship between Age and Height.**
- b) Write the R Script to generate Scatter plot matrix to show relationship between 5 variables which are present in your dataset.**
- c) Write the R script to show mark distribution of students in 3 different sections using Box Plot.**

Procedure:

- **Scatter Plot:** It is used to represent the relationship between two variables in a data-set. Generally the independent variable is plotted on the X-axis, while the dependent variable is plotted on the Y-axis. The simple scatterplot is created using the `plot()` function.

Syntax: `plot(x, y, main, xlab, ylab, xlim, ylim, axes)`

Where

- **x - the data set whose values are the horizontal coordinates.**
- **y - the data set whose values are the vertical coordinates.**
- **main - the title of the graph.**
- **xlab - the label in the horizontal axis.**
- **ylab - the label in the vertical axis.**
- **xlim - the limits of the values of x used for plotting.**
- **ylim - the limits of the values of y used for plotting.**
- **axes - indicates whether both axes should be drawn on the plot.**

- **Scatter Plot Matrix:** When we have more than two variables and we want to find the correlation between one variable versus the remaining ones we use scatterplot matrix. `pairs()` function to create matrices of scatterplots.

Syntax: `pairs(formula, data)`

Where

- formula - the series of variables used in pairs.
- data - the data set from which the variables will be taken.

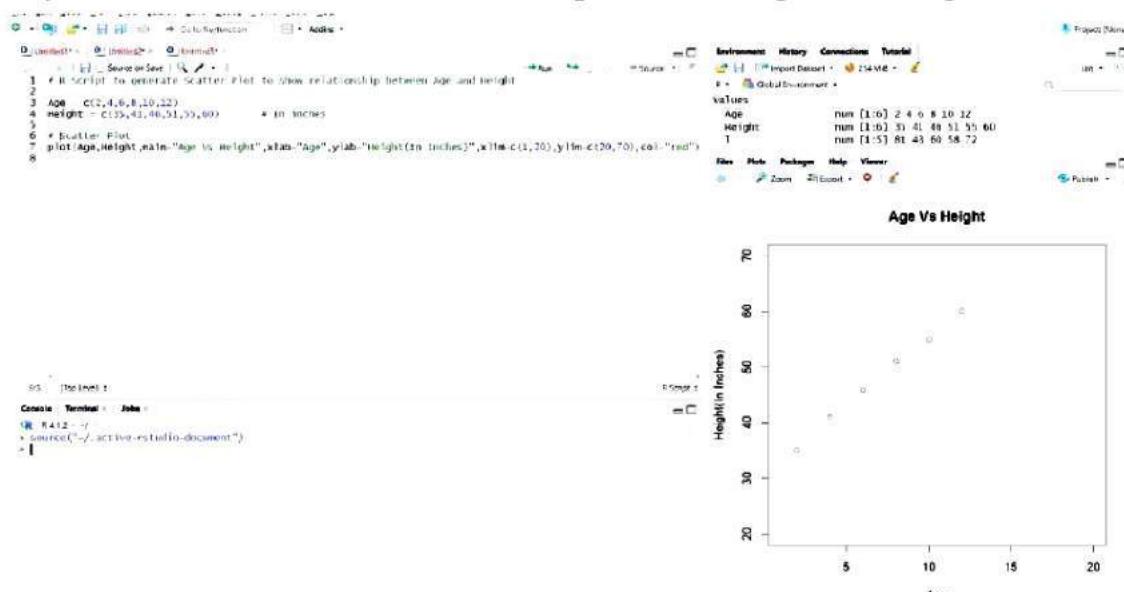
- **Box Plot:** used to display information in the form of distribution by drawing boxplots for each of them. This distribution of data based on five sets (minimum, first quartile, median, third quartile, maximum).

Syntax: `boxplot(x, data, notch, varwidth, names, main, xlab, ylab)`

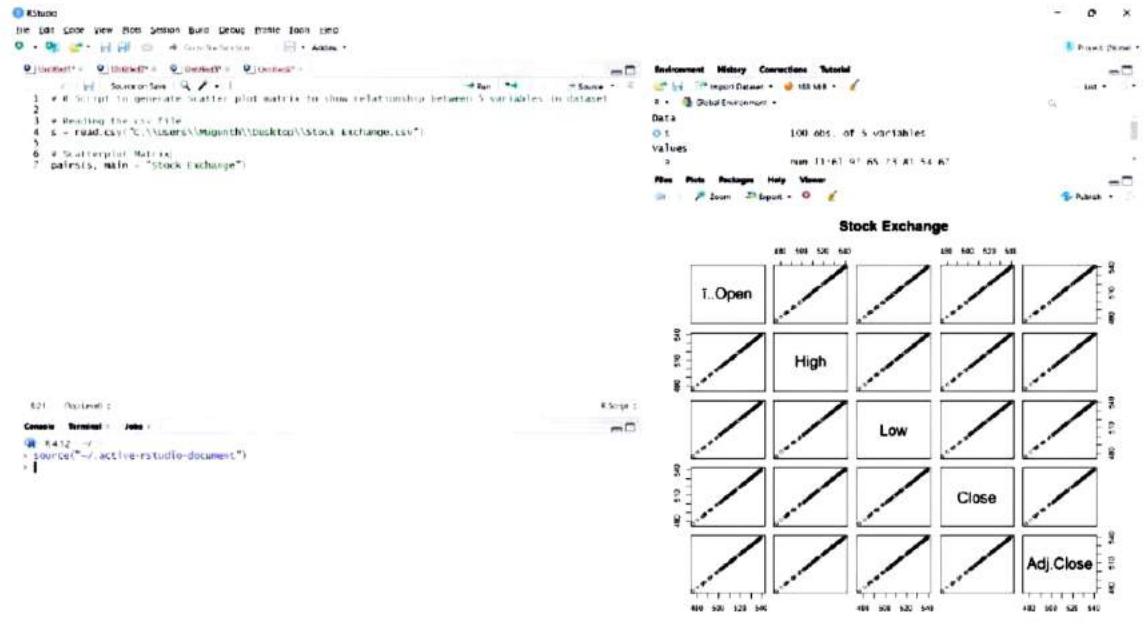
Where

- x - vector or a formula.
- data - data frame.
- notch - a logical value. Set as TRUE to draw a notch.
- varwidth - a logical value. Set as TRUE to draw width of the box proportionate to the sample size.
- names - the group labels which will be printed under each boxplot.
- main - used to give a title to the graph.
- xlab - the label in the horizontal axis.
- ylab - the label in the vertical axis.

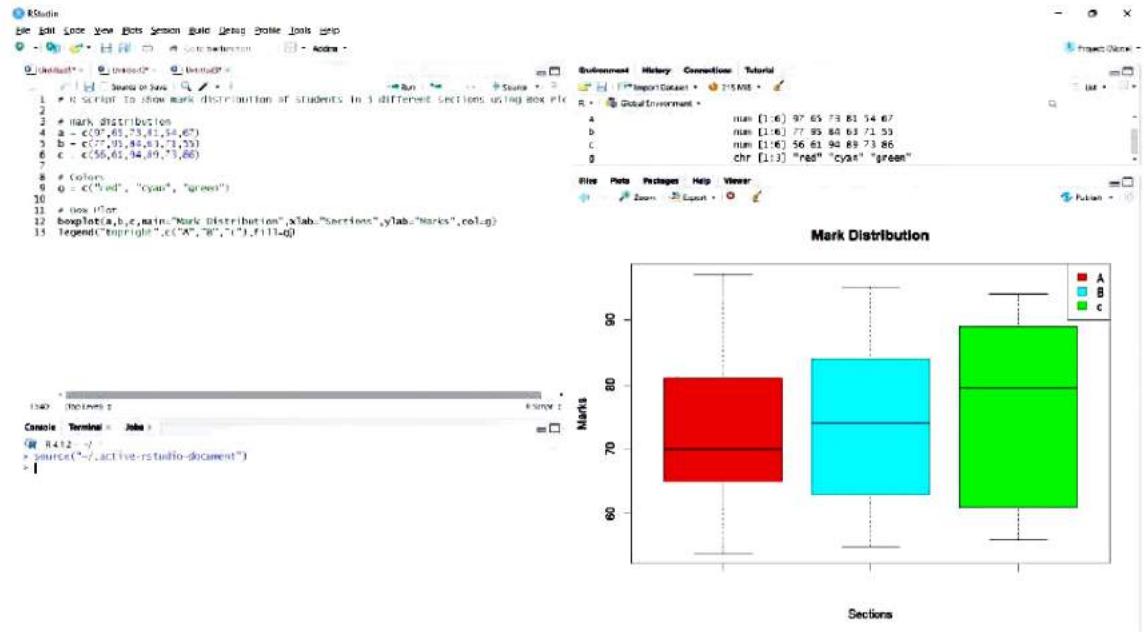
a) Scatter Plot to show relationship between Age and Height.



b) Scatter Plot matrix to show relationship between 5 variables.



c) Box Plot to show mark distribution of students in 3 different sections.



Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 21-01-2021

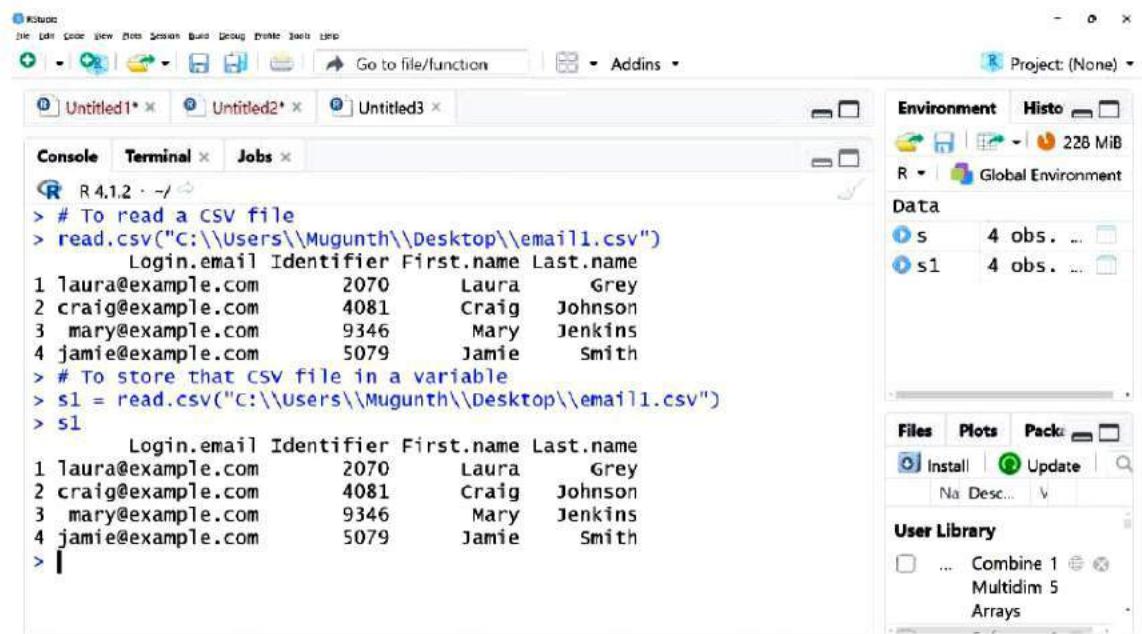
EXPERIMENT NAME: R Script for Reading and writing CSV

Aim:

To read the csv file and write the contents of same into another file and check the same in R.

Procedure:

- Reading the csv file and assigning it to a variable**



The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help, and Help. The top toolbar has icons for file operations like Open, Save, and Print. The title bar says "RStudio" and "Project: (None)". The main window has three tabs in the top bar: "Untitled1*", "Untitled2*", and "Untitled3*". The "Console" tab is active, showing R code and its output. The code reads a CSV file and stores it in a variable s1. The output shows the data frame structure with columns: Login.email, Identifier, First.name, and Last.name, containing four rows of data. To the right of the console are the "Environment", "Data", "Files", and "User Library" panes. The "Environment" pane shows objects s and s1. The "Data" pane shows the same data frame s. The "Files" pane shows the current project structure, and the "User Library" pane shows available packages.

```
R 4.1.2 - ~/  
> # To read a CSV file  
> read.csv("C:\\\\Users\\\\Mugunth\\\\Desktop\\\\email1.csv")  
  Login.email Identifier First.name Last.name  
1 laura@example.com      2070    Laura     Grey  
2 craig@example.com      4081    Craig   Johnson  
3 mary@example.com       9346    Mary    Jenkins  
4 jamie@example.com      5079   Jamie    Smith  
> # To store that CSV file in a variable  
> s1 = read.csv("C:\\\\Users\\\\Mugunth\\\\Desktop\\\\email1.csv")  
> s1  
  Login.email Identifier First.name Last.name  
1 laura@example.com      2070    Laura     Grey  
2 craig@example.com      4081    Craig   Johnson  
3 mary@example.com       9346    Mary    Jenkins  
4 jamie@example.com      5079   Jamie    Smith
```

- Writing the contents of csv file into a new csv file

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Source, View, Tools, Session, Build, Debug, Profile, Tools, Help. The toolbar has icons for New Project, Open, Save, Print, Find, Go to file/function, and Addins. The project bar shows three files: Untitled1*, Untitled2*, and Untitled3*. The left sidebar has tabs for Environment, Terminal, and Jobs. The Terminal tab is active, showing R 4.1.2 running on a Mac OS X system. The user has run the following commands:

```
R 4.1.2 · ~/Documents/RStudio
> # writing the contents of email1.csv into a new csv file
> write.csv(s1, "Random1.csv")
>
```

The Environment pane shows the Global Environment with objects s (4 obs.) and s1 (4 obs.). The User Library pane shows Combine 1, Multidim 5, and Arrays.

- **Reading the new csv file**

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 25-01-2022

EXPERIMENT NAME: R Script for Reading XML Contents

Aims

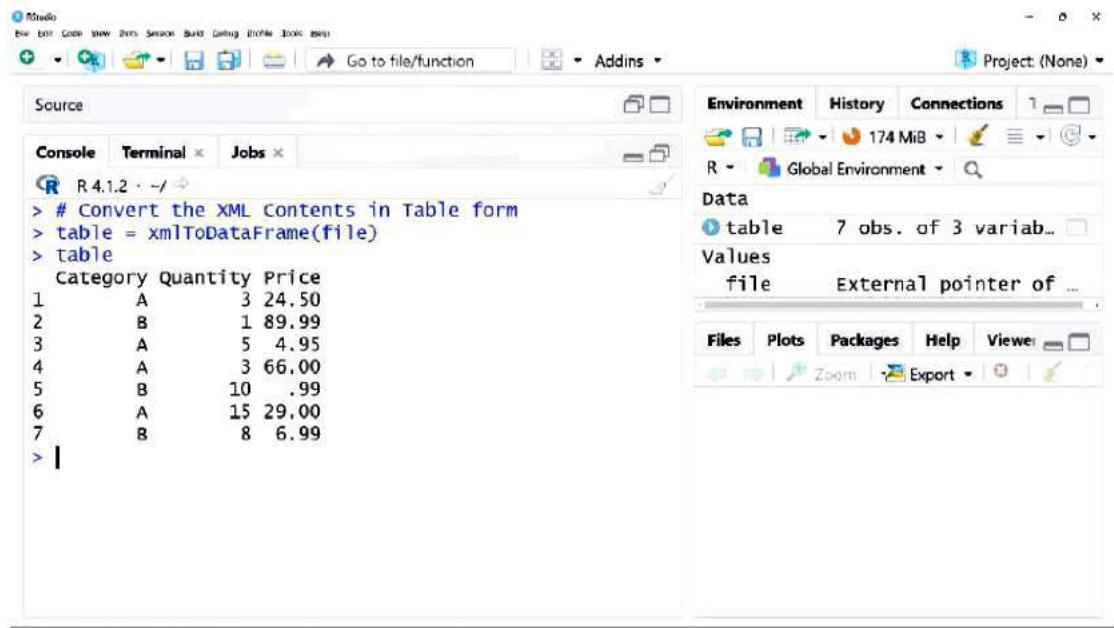
To

- a) Write the R Script to read the attached XML file
 - b) Convert the XML Contents in Table form.
 - c) Extract and display Category A product details.
 - d) Find the total Quantity.
 - e) Display Price Column Alone.
 - f) Extract 4th,5th and 6th rows alone.

Procedure:

a) R Script to read the attached XML file

b) Convert the XML Contents in Table form



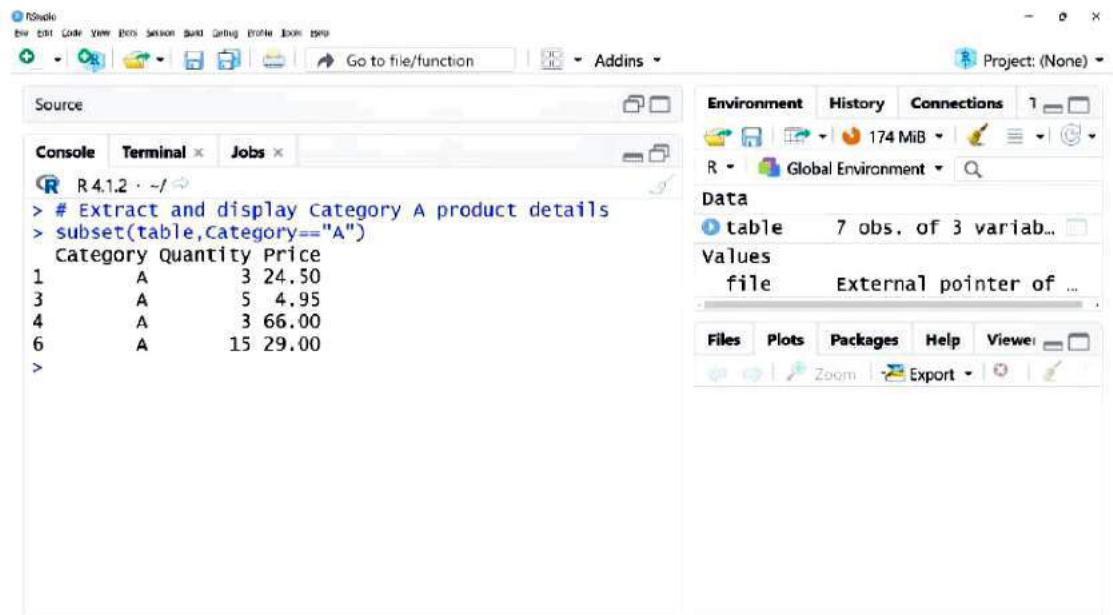
The screenshot shows the RStudio interface. In the Source pane, the following R code is run:

```
R 4.1.2 · ~/ 
> # Convert the XML Contents in Table form
> table = xmlToDataFrame(file)
> table
```

The resulting table is displayed in the Environment pane:

	Category	Quantity	Price
1	A	3	24.50
2	B	1	89.99
3	A	5	4.95
4	A	3	66.00
5	B	10	.99
6	A	15	29.00
7	B	8	6.99

c) Extract and display Category A product details



The screenshot shows the RStudio interface. In the Source pane, the following R code is run:

```
R 4.1.2 · ~/ 
> # Extract and display Category A product details
> subset(table,Category=="A")
```

The resulting table is displayed in the Environment pane:

	Category	Quantity	Price
1	A	3	24.50
3	A	5	4.95
4	A	3	66.00
6	A	15	29.00

d) Find the total Quantity

The screenshot shows the RStudio interface. In the Console tab, the following R code is run:

```
R 4.1.2 · ~/ ~
> # Find the total quantity
> sum(as.integer(table$Quantity))
[1] 45
> |
```

The Environment pane shows a table object with 7 observations and 3 variables. The Values pane shows an external pointer named 'file'.

e) Display Price Column Alone

The screenshot shows the RStudio interface. In the Console tab, the following R code is run:

```
R 4.1.2 · ~/ ~
> # Display Price Column Alone
> table[,3]
[1] "24.50" "89.99" "4.95"  "66.00" ".99"   "29.00"
[7] "6.99"
> |
```

The Environment pane shows a table object with 7 observations and 3 variables. The Values pane shows an external pointer named 'file'.

f) Extract 4th,5th and 6th rows alone

The screenshot shows the RStudio interface. In the top-left, the menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with various icons. The main area has tabs for Source, Console, Terminal, and Jobs. The Console tab is active, displaying R code and its output:

```
R 4.1.2 -->
> # Extract 4th,5th and 6th rows alone
> table[4:6,]
  Category Quantity Price
4       A         3  66.00
5       B        10   .99
6       A        15 29.00
> |
```

To the right of the console is the Environment pane, which shows a table object with 7 observations and 3 variables. The Data pane indicates that the table is an external pointer. The bottom navigation bar includes Files, Plots, Packages, Help, and Viewer.

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 28-01-2022

EXPERIMENT NAME: R Script for Handling Missing Values (Data Preprocessing)

Aim:

To

- a) Download the data set with missing values.**
- b) Check the presence of missing values.**
- c) Calculate mean value with and without missing value.**
- d) Replace missing values with the mean value.**
- e) Display the updated data set.**

Procedure:

a) Download and Read the data set with missing values

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, View, Help, Session, Basic, Debug, Tools, and Help. The main area has tabs for Source, Console, Terminal, and Jobs. The Console tab shows R code and its output. The Environment tab shows a global environment with 100 observations and 5 variables. The Packages tab lists several packages from the user library, including abind, askpass, assertthat, backports, base64enc, bit, and bit64. The bottom status bar indicates the version is R 4.1.2.

```
R 4.1.2 -->
> # Downloading and Reading the data set with missing values
> # Dataset - Stock Exchange.csv
>
> s = read.csv("C:\\users\\mugunth\\Downloads\\Stock_Exchange.csv")
> s
  1..Open  High   Low Close Adj.Close
1 528.69 528.69 528.69 528.69      528.69
2 527.21 527.21 527.21 527.21      527.21
3 527.84 527.84 527.84 527.84      527.84
4    NA 531.12 531.12 531.12      531.12
5 532.07 532.07 532.07 532.07      532.07
6 532.60 532.60 532.60 532.60      532.60
7 533.87 533.87 533.87 533.87      533.87
8 534.29 534.29 534.29 534.29      534.29
9 533.34 533.34 533.34 533.34      533.34
10 534.40 534.40 534.40    NA      534.40
11 535.45 535.45 535.45 535.45      535.45
12 537.46 537.46 537.46 537.46      537.46
13 538.94 538.94 538.94 538.94      538.94
14 537.67 537.67 537.67 537.67      537.67
15 535.67 535.67 535.67 535.67      535.67
16 535.98 535.98 535.98 535.98      535.98
17 537.67    NA 537.67 537.67      537.67
18 538.10 538.10 538.10 538.10      538.10
19 537.57 537.57 537.57    NA      537.57
20 537.36 537.36 537.36 537.36      537.36
21 535.67 535.67 535.67 535.67      535.67
22 533.34 533.34 533.34 533.34      533.34
23 529.11 529.11 529.11 529.11      529.11
24 531.44 531.44 531.44 531.44      531.44
```

b) Check the presence of missing values

The screenshot shows the RStudio interface. In the top-left, the menu bar includes File, Edit, Code, View, Data, Session, Build, Debug, Tools, Help, and Tools. Below the menu is a toolbar with icons for file operations like Open, Save, Print, and a search bar labeled 'Go to file location'. The main area has tabs for 'Source', 'Console' (active), 'Terminal', and 'Jobs'. The console window displays R code and its output:

```
R 4.1.2 -->
> # check the presence of Missing values
> is.na(s)
#> [1] Open High Low Close Adj.Close
#> [2,] FALSE FALSE FALSE FALSE FALSE
#> [3,] FALSE FALSE FALSE FALSE FALSE
#> [4,] TRUE FALSE FALSE FALSE FALSE
#> [5,] FALSE FALSE FALSE FALSE FALSE
#> [6,] FALSE FALSE FALSE FALSE FALSE
#> [7,] FALSE FALSE FALSE FALSE FALSE
#> [8,] FALSE FALSE FALSE FALSE FALSE
#> [9,] FALSE FALSE FALSE FALSE FALSE
#> [10,] FALSE FALSE FALSE TRUE FALSE
#> [11,] FALSE FALSE FALSE FALSE FALSE
#> [12,] FALSE FALSE FALSE FALSE FALSE
#> [13,] FALSE FALSE FALSE FALSE FALSE
#> [14,] FALSE FALSE FALSE FALSE FALSE
#> [15,] FALSE FALSE FALSE FALSE FALSE
#> [16,] FALSE FALSE FALSE FALSE FALSE
#> [17,] FALSE TRUE FALSE FALSE FALSE
#> [18,] FALSE FALSE FALSE FALSE FALSE
#> [19,] FALSE FALSE FALSE TRUE FALSE
#> [20,] FALSE FALSE FALSE FALSE FALSE
#> [21,] FALSE FALSE FALSE FALSE FALSE
#> [22,] FALSE FALSE FALSE FALSE TRUE
#> [23,] FALSE FALSE FALSE FALSE FALSE
#> [24,] FALSE FALSE FALSE FALSE FALSE
#> [25,] FALSE FALSE TRUE FALSE FALSE
#> [26,] FALSE FALSE FALSE FALSE FALSE
#> [27,] TRUE FALSE FALSE FALSE FALSE
#> [28,] ...
```

To the right of the console is the 'Environment' tab of the viewer pane, showing a global environment with one object 's' containing 100 observations of 5 variables. Below the viewer is a package browser titled 'User Library' with a list of available packages.

c) Calculate mean value with and without missing value

The screenshot shows the RStudio interface. The menu bar and toolbar are identical to the previous screenshot. The console window displays R code and its output:

```
R 4.1.2 -->
> # Calculate mean value with missing value
> openmean = mean(s$1,.Open)
> openmean
[1] NA
> # Calculate mean value without missing value
> openmean = mean(s$1,.Open,na.rm = TRUE)
> openmean
[1] 520.1641
>
```

The 'Environment' tab of the viewer pane shows the variable 'openmean' with a value of 520.1641. The 'Data' tab shows the same information as before. The package browser 'User Library' is also present at the bottom.

d) Replace missing values with the mean value

R 4.1.2 - ~/

```
# Replace missing values with the mean value
> s1<-Open = ifelse(is.na(s1[,Open]), Openmean, s1[,Open])
> s1[,Open]
#> [1] 528.6000 527.2100 527.8400 520.1641 522.0700 523.6000 523.8700 524.2000
#> [9] 523.3400 524.4000 525.4500 527.4600 528.9400 527.6700 525.6700 521.9800
#> [17] 527.6700 528.1000 527.5700 527.3600 525.6700 523.1400 529.1100 521.4400
#> [25] 522.0700 523.4500 520.1641 526.8100 529.8900 528.4100 528.3100 527.7800
#> [33] 525.4500 525.3100 522.4900 521.4400 528.2800 525.9400 522.4500 523.9300
#> [41] 524.5600 527.9000 512.8300 514.9400 515.3500 506.3800 525.9000 511.1300
#> [49] 511.2400 510.6200 504.9000 503.5100 504.2600 505.6500 520.5541 512.1900
#> [57] 513.6700 511.9800 512.7200 513.9900 514.8400 512.8300 501.6000 512.3500
#> [65] 515.8800 510.2300 523.5300 524.8800 525.9400 525.7300 524.1400 524.3500
#> [73] 525.4100 526.2600 524.2500 524.5600 527.4200 529.1100 520.0500 527.5200
#> [81] 526.1500 524.3500 521.1800 520.1641 520.5400 514.7300 511.9800 503.1000
#> [89] 502.1500 493.4800 497.7100 498.5500 492.4200 487.3400 480.8900 476.4500
#> [97] 485.6000 484.8100 487.1300 491.4700
```

> s

	Open	High	Low	Close	Adj.Close
1	528.6000	528.69	528.69	528.69	528.69
2	527.2100	527.21	527.21	527.21	527.21
3	527.8400	527.84	527.84	527.84	527.84
4	520.1641	521.12	521.12	521.12	521.12
5	522.0700	522.07	522.07	522.07	522.07
6	523.6000	522.66	522.66	522.60	522.60
7	523.8700	523.87	523.87	523.87	523.87
8	524.2000	524.29	524.29	524.29	524.29
9	523.3400	523.34	523.34	523.34	523.34
10	524.4000	524.40	524.40	524.40	524.40
11	525.4500	525.45	525.45	525.45	525.45
12	527.4600	527.46	527.46	527.46	527.46
13	526.9400	526.94	526.94	526.94	526.94
14	527.6700	527.67	527.67	527.67	527.67
15	525.6700	525.67	525.67	525.67	525.67
16	525.9800	525.98	525.98	525.98	525.98
17	527.6700	NA	527.67	527.67	527.67

e) Calculating the mean values without missing values and replacing the missing values with mean values in other columns

R 4.1.2 - ~/

```
# Calculating the mean values without missing values and Replacing the
# missing values with mean values
> Highmean = mean(s$High, na.rm = TRUE)
> Lowmean = mean(s$Low, na.rm = TRUE)
> Closemean = mean(s$Close, na.rm = TRUE)
> Adjclosemean = mean(s$Adj.Close, na.rm = TRUE)
> s$High = ifelse(is.na(s$High), Highmean, s$High)
> s$Low = ifelse(is.na(s$Low), Lowmean, s$Low)
> s$Close = ifelse(is.na(s$Close), Closemean, s$Close)
> s$Adj.Close = ifelse(is.na(s$Adj.Close), Adjclosemean, s$Adj.Close)
> |
```

f) Checking for any missing data after Data Preprocessing

The screenshot shows the RStudio interface. In the top-left, the menu bar includes File, Edit, View, Data, Session, Build, Debug, Tools, Help, and Help + Topics. The top-right has tabs for Environment, History, Connections, and Tutorial, with 'Environment' selected. The main workspace shows a data frame 's' with 100 rows and 5 columns. The columns are labeled 'AdjClosemean', 'Closemean', 'Highmean', 'Lowmean', and 'Openmean'. The bottom part of the screen displays the 'gridExtra' package in the 'User Library' section of the package browser.

g) Display the updated data set

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 08-02-2022

EXPERIMENT NAME: Statistics using R

Aim:

To

- a) Display structure of mtcars data set(already available in R Studio).
- b) Find mean value of mpg attribute.
- c) Find the median value of Wt Attribute.
- d) Find the mode value of cyl attribute.
- e) Find the Quartiles of hp attribute.

Procedure:

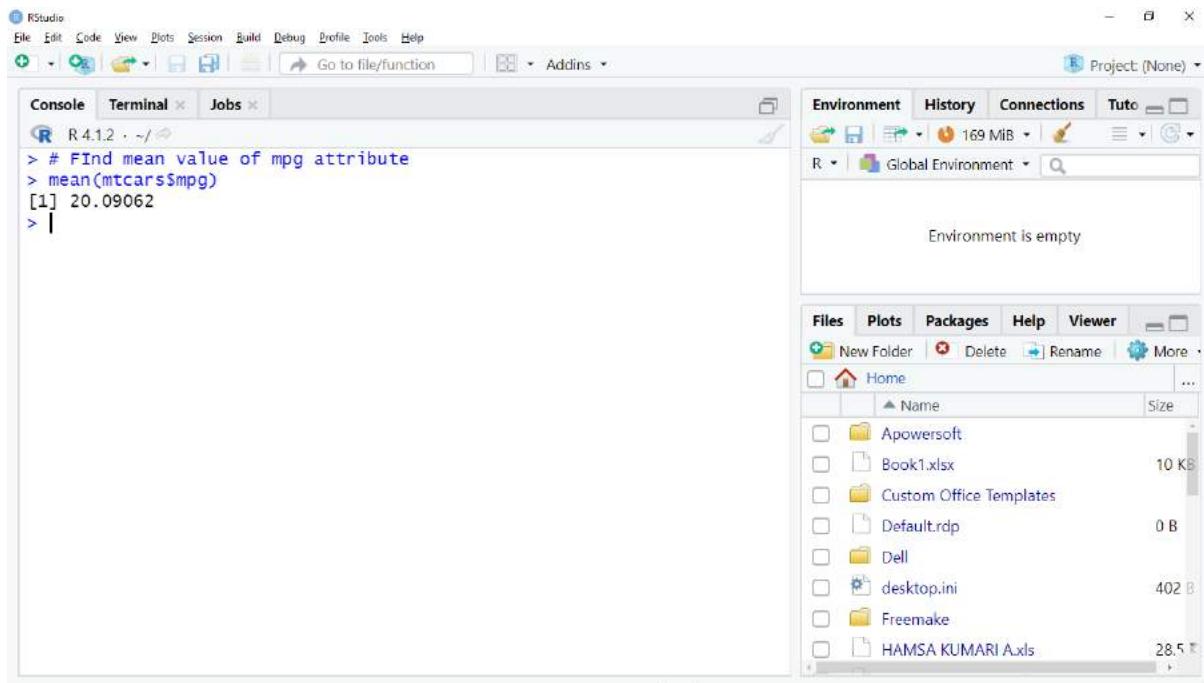
a) R Script to display structure of mtcars data set

The screenshot shows the RStudio interface. The R console pane displays the following R script and its output:

```
R 4.1.2 · ~/Desktop
> # Display structure of mtcars data set
> str(mtcars)
'data.frame': 32 obs. of 11 variables:
 $ mpg : num 21 21 22.8 21.4 18.7 ...
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num 160 160 108 258 360 ...
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num 16.5 17 18.6 19.4 17 ...
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num 4 4 4 3 3 3 4 4 4 ...
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

The environment pane shows that the Global Environment is empty. The files pane lists several files and folders on the desktop, including 'Apowersoft', 'Book1.xlsx', 'Custom Office Templates', 'Default.rdp', 'Dell', 'desktop.ini', 'Freemake', and 'HAMSA KUMARI Axls'. The plots pane is currently empty.

b) R Script to find mean value of mpg attribute



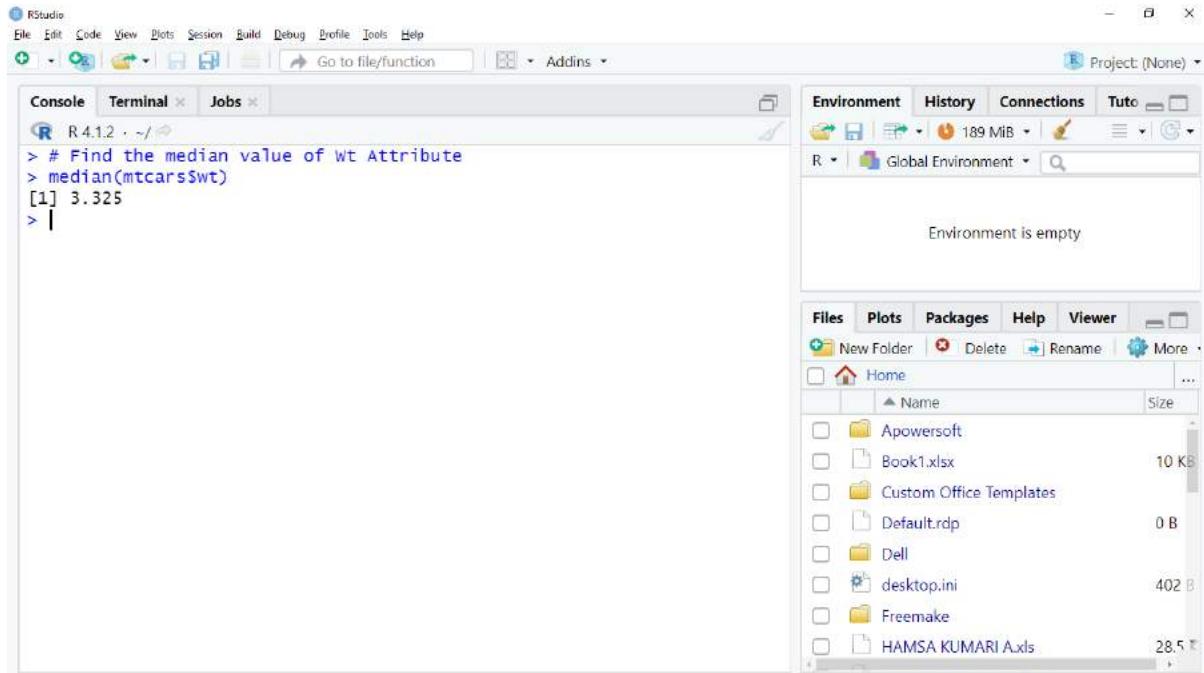
The screenshot shows the RStudio interface. The console tab is active, displaying the following R code and output:

```
R 4.1.2 · ~/Desktop
> # Find mean value of mpg attribute
> mean(mtcars$mpg)
[1] 20.09062
> |
```

The environment pane shows the global environment is empty. The files pane displays several files and folders in the current directory:

Name	Size
Apowersoft	10 KB
Book1.xlsx	
Custom Office Templates	
Default.rdp	0 B
Dell	
desktop.ini	402 B
Freemake	
HAMSA KUMARI Axls	28.5 KB

c) R script to find the median value of Wt Attribute



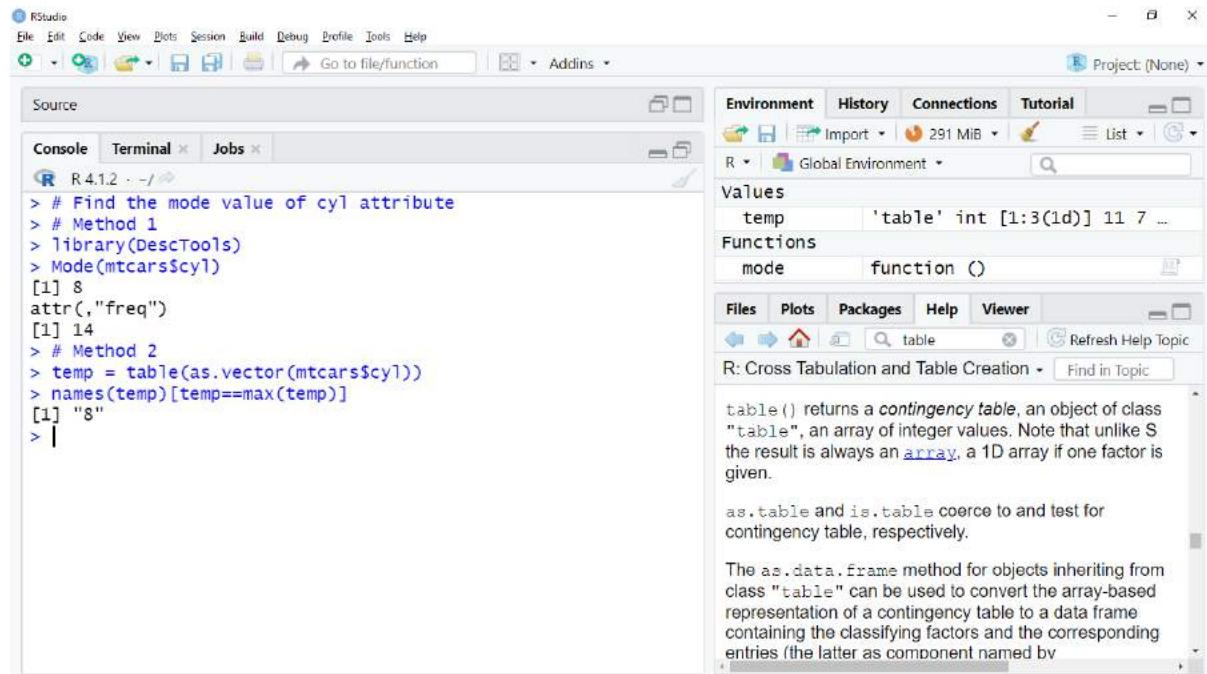
The screenshot shows the RStudio interface. The console tab is active, displaying the following R code and output:

```
R 4.1.2 · ~/Desktop
> # Find the median value of Wt Attribute
> median(mtcars$wt)
[1] 3.325
> |
```

The environment pane shows the global environment is empty. The files pane displays several files and folders in the current directory:

Name	Size
Apowersoft	10 KB
Book1.xlsx	
Custom Office Templates	
Default.rdp	0 B
Dell	
desktop.ini	402 B
Freemake	
HAMSA KUMARI Axls	28.5 KB

d) R script to find the mode value of cyl attribute



RStudio interface showing the mode value of cyl attribute found via library(DescTools). The console output is as follows:

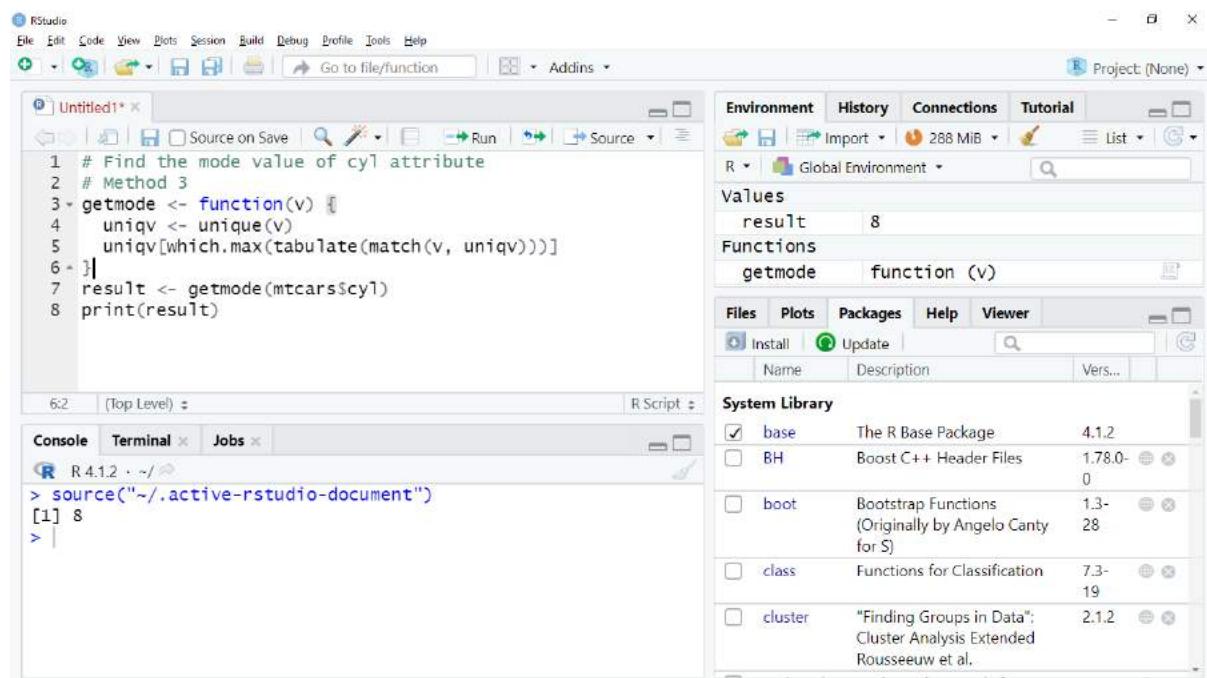
```
> # Find the mode value of cyl attribute
> # Method 1
> library(DescTools)
> Mode(mtcars$cyl)
[1] 8
attr(,"freq")
[1] 14
> # Method 2
> temp = table(as.vector(mtcars$cyl))
> names(temp)[temp==max(temp)]
[1] "8"
>
```

The Global Environment pane shows a table object named 'temp' with values [1:3(1d)] 11 7 ...

The Help pane provides information about the table() function, stating it returns a contingency table, an object of class "table", an array of integer values. Note that unlike S the result is always an array, a 1D array if one factor is given.

The as.table and is.table coerce to and test for contingency table, respectively.

The as.data.frame method for objects inheriting from class "table" can be used to convert the array-based representation of a contingency table to a data frame containing the classifying factors and the corresponding entries (the latter as component named by



RStudio interface showing the mode value of cyl attribute found via a custom getmode function. The code in the script editor is:

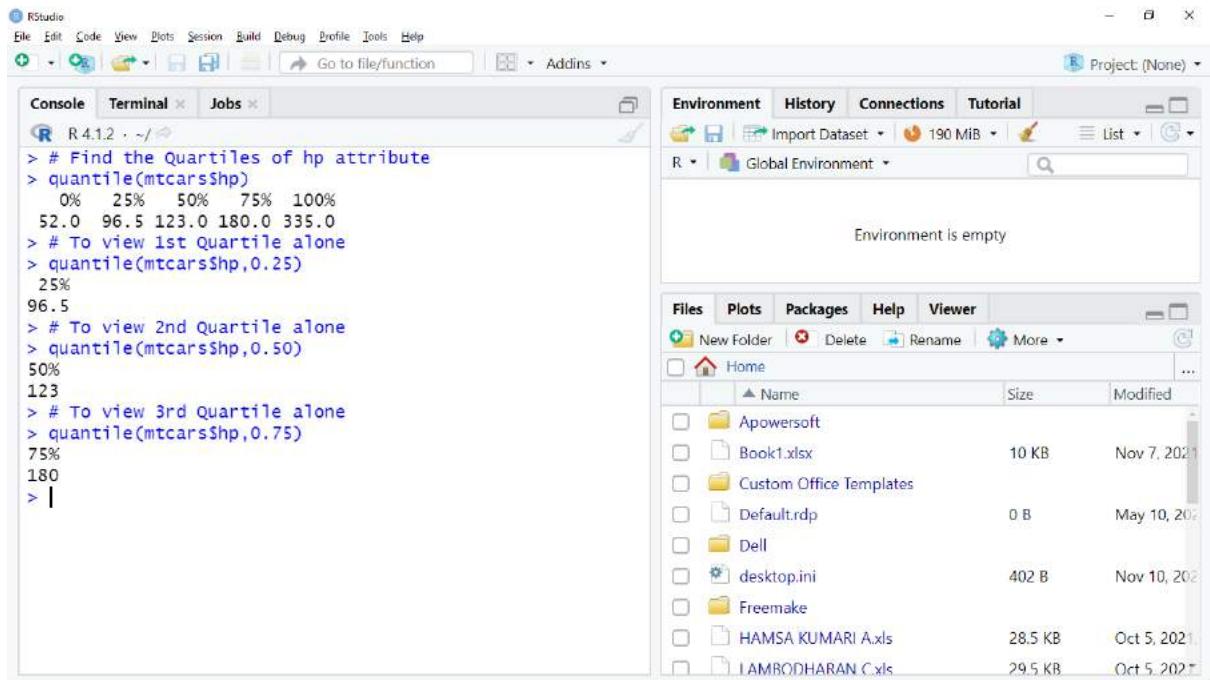
```
1 # Find the mode value of cyl attribute
2 # Method 3
3 getmode <- function(v) {
4   uniqv <- unique(v)
5   uniqv[which.max(tabulate(match(v, uniqv)))]
6 }
7 result <- getmode(mtcars$cyl)
8 print(result)
```

The Global Environment pane shows a result object with value 8.

The System Library pane lists the following packages:

Name	Description	Vers...
base	The R Base Package	4.1.2
BH	Boost C++ Header Files	1.78.0-0
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
class	Functions for Classification	7.3-19
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.1.2

e) R script to find the Quartiles of hp attribute



The screenshot shows the RStudio interface with the following details:

- Console Tab:** Displays the R code and its output. The code uses the `quantile` function on the `mtcars\$hp` column to find quartiles. The output shows the minimum, Q1, median, Q3, and maximum values.
- Environment Tab:** Shows the Global Environment pane with the message "Environment is empty".
- Files Tab:** Shows a list of files in the current directory, including various Excel and Word documents.

```
R 4.1.2 · ~/Desktop
> # Find the Quartiles of hp attribute
> quantile(mtcars$hp)
  0%   25%   50%   75% 100%
52.0  96.5 123.0 180.0 335.0
> # To view 1st Quartile alone
> quantile(mtcars$hp,0.25)
25%
96.5
> # To view 2nd Quartile alone
> quantile(mtcars$hp,0.50)
50%
123
> # To view 3rd Quartile alone
> quantile(mtcars$hp,0.75)
75%
180
> |
```

Result:

Thus, the program has been executed and the output has been verified.

NAME: Mugunth.D

REG NO: 40733020

SUBJECT: R PROGRAMMING

DATE: 01-03-2022

EXPERIMENT NAME: Machine Learning Algorithms in R

Aim:

To

1.Load the iris data set.

2.Identify the best model to predict Species of given flower based on width of Sepal and Petal and length of Sepal and Petal.

Procedure:

1. Load the data set in RStudio.

2. for creating a model :

We create test and train data by dividing the data set into 2 parts 75% and 25%.

3. we need to choose a model for predictions.

4. we use split() to split the dataset into train and test datasets.

5. We create a model.

6. Train the model using train dataset.

7. Predict values by entering the test dataset to the model.

8. Test the accuracy of the model(using confusion matrix).

Syntax and code of:

1. Decision tree:

We need to install.packages("party")

Library(party)

Syntax: Ctree(formula,data)

The figure shows the RStudio interface with the following components:

- File Edit Code View Plots Session Build Debug Profile Tools Help**: The top menu bar.
- Random forest.R Decision Tree SVM.R**: The project navigation bar.
- Source on Save**: A button in the top right corner.
- R Script**: The main code editor window containing the R script for building a decision tree.
- Console**: The terminal window showing the execution of the R script and the resulting output.
- Environment**, **History**, **Connections**, **Tutorial**: The top navigation tabs for the RStudio environment.
- Plots**: The active tab in the top navigation bar.
- Decision Tree Plot**: A plot showing a decision tree for the Iris dataset. The root node (Node 1) splits on Petal.Length (< 1.9 vs > 1.9). The left branch leads to Node 2 (n=40), which is a leaf node labeled "setosa". The right branch leads to Node 3 (n=35), which further splits on Petal.Width (< 1.7 vs > 1.7). Node 3 leads to Node 4 (n=8), which splits on Petal.Length (< 4.7 vs > 4.7). Node 4 leads to Node 5 (n=35), which is a leaf node labeled "setosa". Node 4 also leads to Node 6 (n=8), which is a leaf node labeled "setosa". Node 3 also leads to Node 7 (n=37), which is a leaf node labeled "setosa".
- Legend**: Below the plot, there are four bar charts representing the class distribution at each node. Each chart has "setosa" on the x-axis and proportions from 0 to 1 on the y-axis. Node 2 (n=40) shows 100% setosa. Node 5 (n=35) shows 100% setosa. Node 6 (n=8) shows approximately 75% setosa and 25% virginica. Node 7 (n=37) shows approximately 95% setosa and 5% virginica.

2. Random Forest:

We need to install.packages("RandomForest")

Library(RandomForest)

Syntax: randomForest(formula,data)

The screenshot shows the RStudio interface with the following details:

- File**: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help
- Project**: RandomForest.Rproj
- Code Editor**: RandomForest.R (Session Tree.R, SVM.R)
 - Code:

```
1 # Random Forest
2
3 library(caret)
4 s = sample.split(iris, splitratio = .8)
5 s
6 train = subset(iris, s == TRUE)
7 test = subset(iris, s == FALSE)
8 dim(train)
9 dim(test)
10 library(randomForest)
11 m = randomForest(Species ~., data = train)
12 m
13 plot(m)
14 p = prediction(m, test)
15 print(p)
16 a = addmargins(table(p, test$Species))
17 print(a)
18 > Accuracy = ((a[1,1]+a[2,2]+a[3,3])/(a[1,1]+a[1,2]+a[1,3]+a[2,1]+a[2,2]+a[2,3]+a[3,1]+a[3,2]+a[3,3]))*100
19 print(Accuracy)
```
 - Output:

```
18:102 [Top Level] >
Concole Terminal Jobs
R 4.1.2 --> /usr/bin/R M-2021.07.15-0000
50      55      60      65      70      75      80      85      90
setosa versicolor versicolor versicolor versicolor versicolor versicolor versicolor
95      100      105      110      115      120      125      130      135
versicolor versicolor virginica virginica virginica versicolor virginica virginica
140      145      150
virginica virginica virginica
Levels: setosa versicolor virginica
```

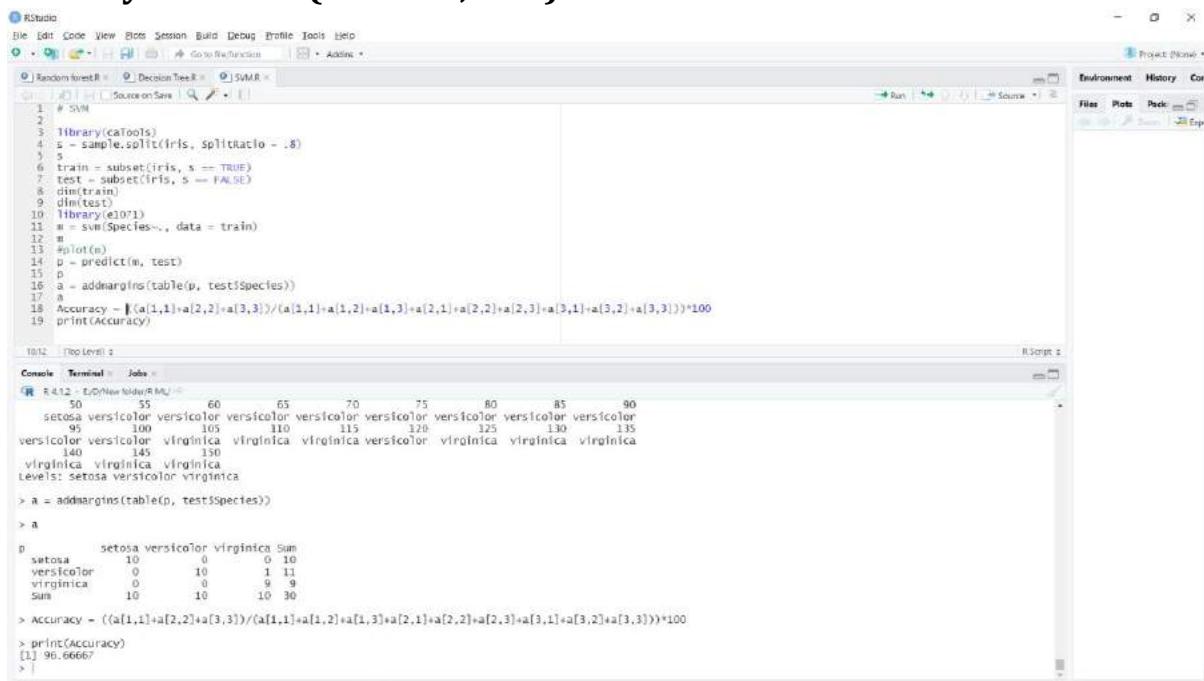
```
> a = addmargins(table(p, test$Species))
> print(a)
```

	setosa	versicolor	virginica	Sum
setosa	10	0	0	10
versicolor	0	10	2	12
virginica	0	0	8	8
Sum	10	10	10	30

```
> Accuracy = ((a[1,1]+a[2,2]+a[3,3])/(a[1,1]+a[1,2]+a[1,3]+a[2,1]+a[2,2]+a[2,3]+a[3,1]+a[3,2]+a[3,3]))*100
> print(Accuracy)
[1] 93.33333
> 
```- Environment**: Environment, History, Connections, Tutorial
- Plots**: A box plot titled "m" showing the distribution of error values across 500 trees. The y-axis ranges from 0.00 to 0.06, and the x-axis ranges from 0 to 500.

3. Support Vector Machine (SVM):

We need to install.packages("e1071")
Library(e1071)
Syntax: svm(formula,data)



The screenshot shows the RStudio interface with the following details:

- File menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** New, Open, Save, Run, Source, Environment, History, Packages, Plots, Help, Viewer.
- Project:** Project (None).
- Code Editor:** Shows R code for SVM. It includes loading packages (caTools, e1071), splitting the iris dataset, fitting an SVM model, and calculating accuracy. The output shows the confusion matrix and accuracy (96.66667%).
- Console:** Displays the R session output, including the command history and the resulting accuracy value.
- Environment:** Shows the global environment with objects like f, m, model1, test, train, and x1.

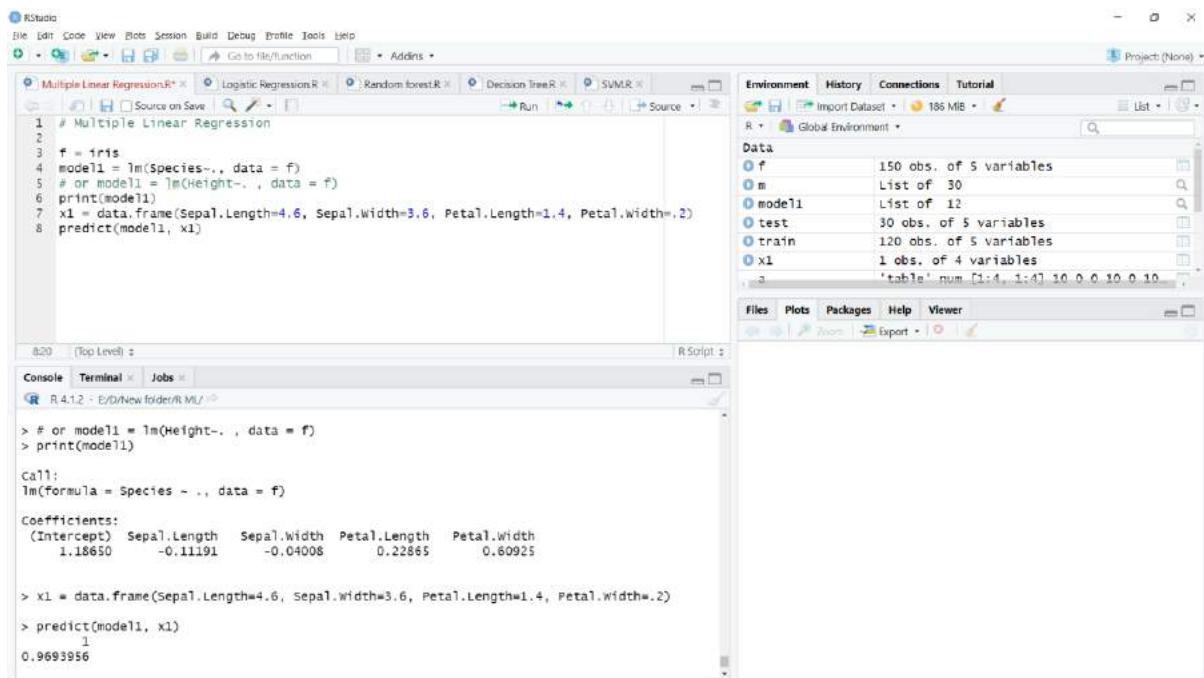
```

1 # SVM
2
3 library(caTools)
4 s = sample.split(iris, splitratio = .8)
5 train = subset(iris, s == TRUE)
6 test = subset(iris, s == FALSE)
7 dim(train)
8 dim(test)
9
10 library(e1071)
11 m = svm(Species~., data = train)
12 n = predict(m)
13 p = predict(m, test)
14 a = addmargins(table(p, test$Species))
15
16 Accuracy = ((a[1,1]+a[2,2]+a[3,3])/(a[1,1]+a[1,2]+a[1,3]+a[2,1]+a[2,2]+a[2,3]+a[3,1]+a[3,2]+a[3,3]))*100
17 print(Accuracy)
18
19 print(Accuracy)
20 [1] 96.66667
21

```

4. Linear Regression:

Using function lm()
Syntax : lm(formula,data)



The screenshot shows the RStudio interface with the following details:

- File menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** New, Open, Save, Run, Source, Environment, History, Connections, Tutorial, Packages, Plots, Help, Viewer.
- Project:** Project (None).
- Code Editor:** Shows R code for multiple linear regression. It loads the iris dataset, fits a model, and makes predictions. The output shows the coefficients and a prediction for a new observation.
- Console:** Displays the R session output, including the command history and the resulting prediction value (0.9693956).
- Environment:** Shows the global environment with objects like f, model1, test, train, and x1.

```

1 # Multiple Linear Regression
2
3 f = iris
4 model1 = lm(Species~., data = f)
5 # or model1 = lm(Height~., data = f)
6 print(model1)
7 x1 = data.frame(Sepal.Length=4.6, Sepal.Width=3.6, Petal.Length=1.4, Petal.Width=.2)
8 predict(model1, x1)

> # or model1 = lm(Height~., data = f)
> print(model1)

Coefficients:
(Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
1.18650    -0.11191   -0.04008    0.22865    0.60925

> x1 = data.frame(Sepal.Length=4.6, Sepal.Width=3.6, Petal.Length=1.4, Petal.Width=.2)
> predict(model1, x1)
1
0.9693956

```

Result:

| Models | SVM | Decision Tree | Random Forest | Linear Regression |
|----------|-------|---------------|---------------|-------------------|
| Accuracy | 96.67 | 93.33 | 96.67 | 96.67 |

From these models we can say that SVM, Decision Tree, Linear Regression, Random Forest gave the good accuracy. Hence we can use either SVM or Decision Tree for predicting the species.

NAME: Mugunth.D

SUBJECT: R PROGRAMMING

REG NO: 40733020

DATE: 22-03-2022

EXPERIMENT NAME: Clustering Algorithms in R

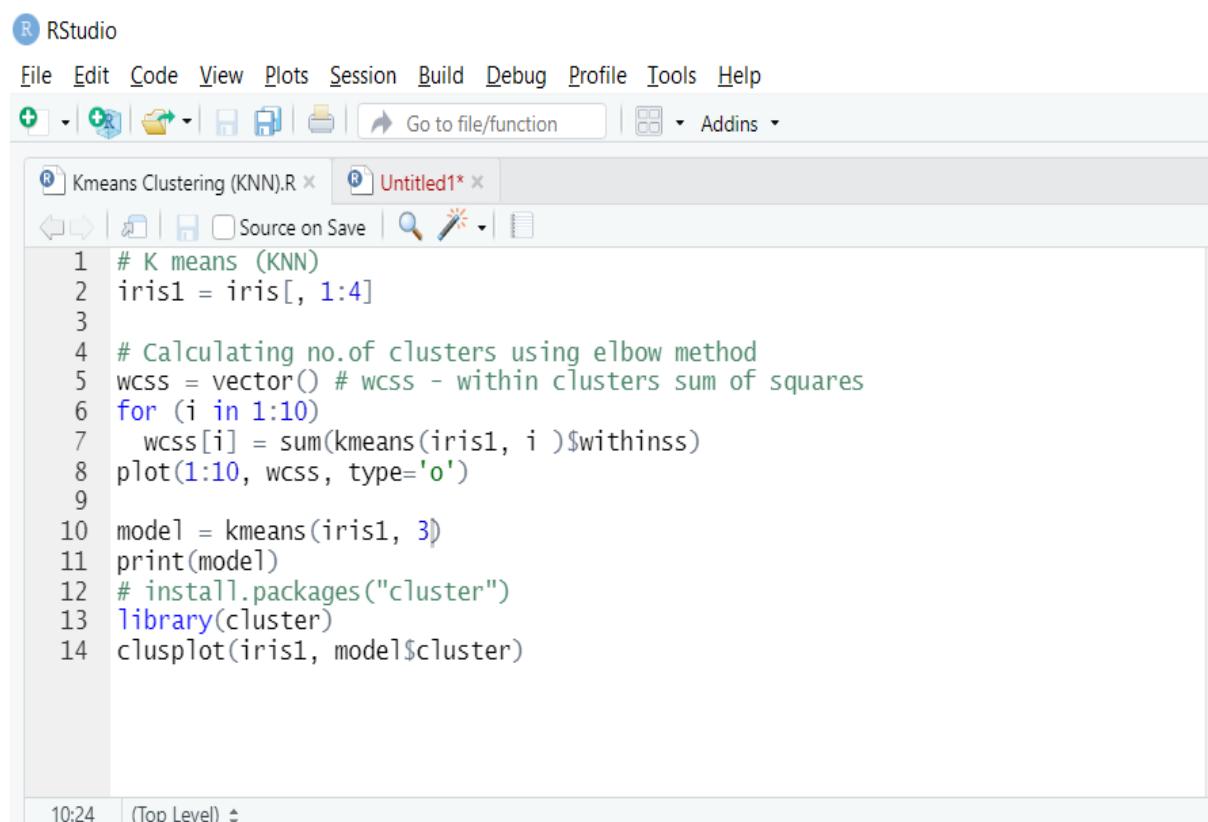
Aim:

To

1. Load the iris data set and USArrests data set.
2. Cluster the data sets using K means Clustering and also find optimal number of clusters using Elbow Method.

Procedure:

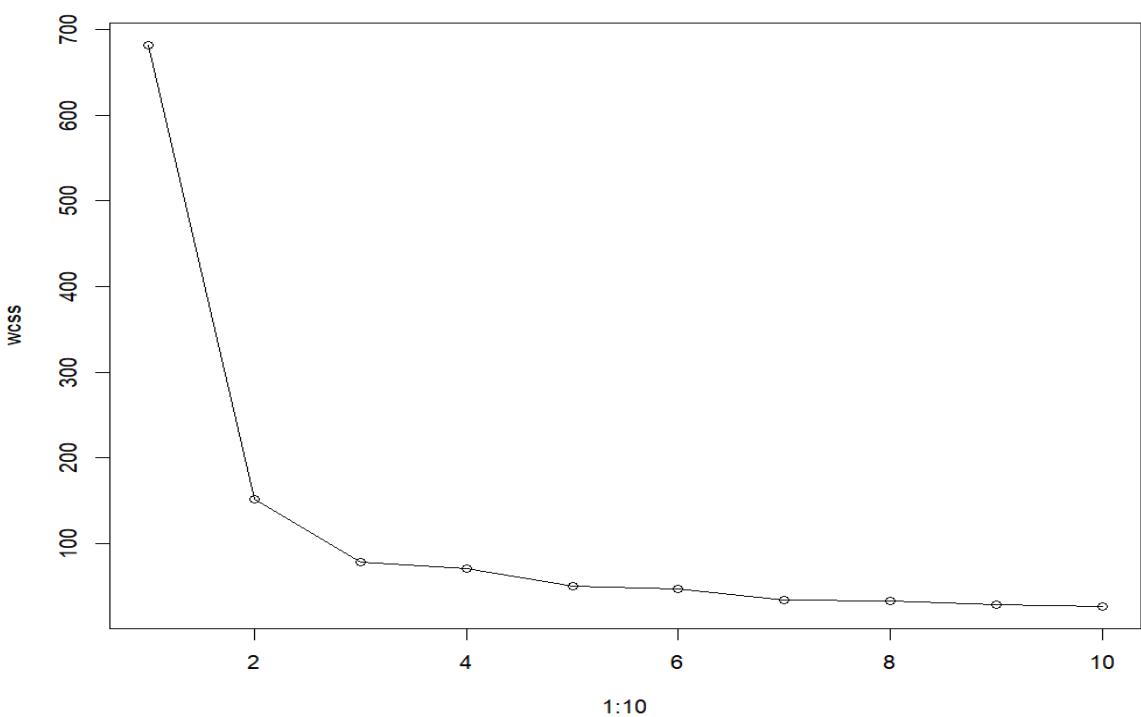
- a) Loading iris data set and clustering the data sets using K means Clustering.

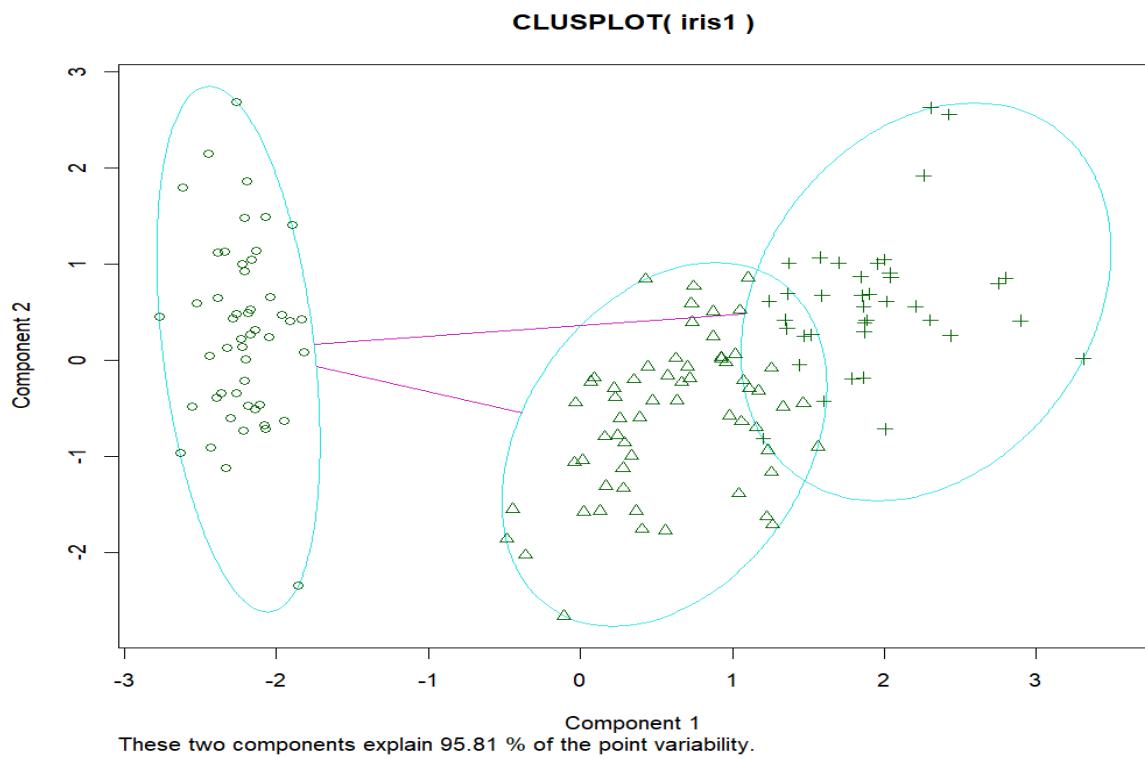


The screenshot shows the RStudio interface with the following details:

- File Menu:** File Edit Code View Plots Session Build Debug Profile Tools Help
- Toolbar:** Includes icons for New Project, Open Project, Save, Print, Go to file/function, and Addins.
- Code Editor:** Displays the R script "Kmeans Clustering (KNN).R".
- Script Content:**

```
1 # K means (KNN)
2 iris1 = iris[, 1:4]
3
4 # Calculating no.of clusters using elbow method
5 wcss = vector() # wcss - within clusters sum of squares
6 for (i in 1:10)
7   wcss[i] = sum(kmeans(iris1, i )$withinss)
8 plot(1:10, wcss, type='o')
9
10 model = kmeans(iris1, 3)
11 print(model)
12 # install.packages("cluster")
13 library(cluster)
14 clusplot(iris1, model$cluster)
```
- Status Bar:** Shows the time as 10:24 and the current level as (Top Level).





b) Loading iris data set and clustering the data sets using K means Clustering.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1* Go to file/function Addins *

Kmeans Clustering (KNN).R

```

1 # K means (KNN)
2 d = USArrests[, 1:4]
3 sd = scale(d) # Scaling the data
4 sd_new = ifelse(sd>.5, 1, 0) # Standardizing the data
5 print(sd_new)
6
7 # Calculating no.of clusters using elbow method
8 wcss = vector() # wcss - within clusters sum of squares
9 for (i in 1:10)
10   wcss[i] = sum(kmeans(sd_new, i )$withinss)
11 plot(1:10, wcss, type='o')
12
13 model = kmeans(sd_new, 4)
14 print(model)
15 # install.packages("cluster")
16 library(cluster)
17 clusplot(sd_new, model$cluster)

```

```

Console Terminal Jobs
R 4.1.2 · ~/rstudio-document
> source("~/active-rstudio-document")
      Murder Assault UrbanPop Rape
Alabama     1       1       0     0
Alaska      1       1       0     1
Arizona     0       1       1     1
Arkansas    0       0       0     0
California   0       1       1     1
Colorado    0       0       1     1
Connecticut  0       0       1     0
Delaware    0       1       0     0
Florida     1       1       1     1
Georgia     1       0       0     0
Hawaii      0       0       1     0
Idaho       0       0       0     0
Illinois    1       1       1     0
Indiana     0       0       0     0
Iowa        0       0       0     0
Kansas      0       0       0     0
Kentucky    0       0       0     0
Louisiana   1       1       0     0
Maine       0       0       0     0
Maryland    1       1       0     1
Massachusetts 0       0       1     0
Michigan    1       1       1     1
Minnesota   0       0       0     0
Mississippi 1       1       0     0
Missouri    0       0       0     1
Montana     0       0       0     0
Nebraska    0       0       0     0
Nevada      1       1       1     1
New Hampshire 0       0       0     0
New Jersey   0       0       1     0
New Mexico   1       1       0     1
New York     1       1       1     1
North Carolina 1       1       0     0
North Dakota 0       0       0     0
Ohio         0       0       1     0
Oklahoma    0       0       0     0
Oregon       0       0       0     1
Pennsylvania 0       0       0     0
Rhode Island 0       0       1     0

```

```

Console Terminal Jobs
R 4.1.2 · ~/rstudio-document
wyoming
K-means clustering with 4 clusters of sizes 10, 23, 7, 10

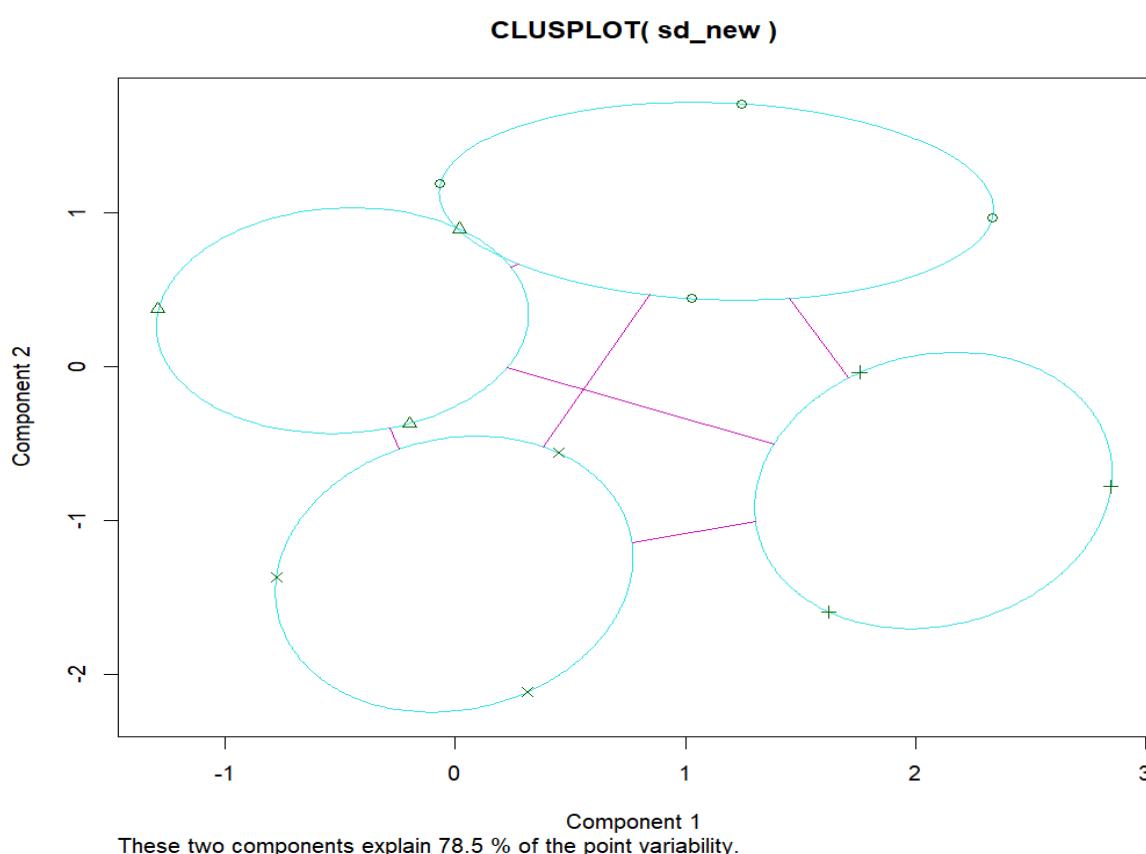
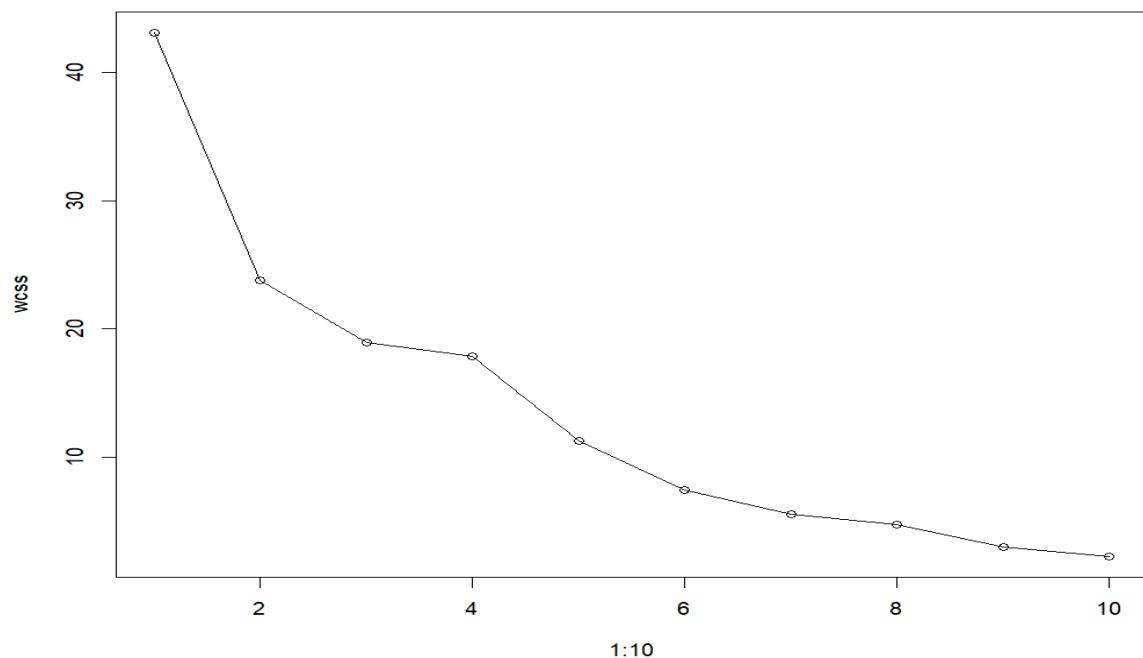
Cluster means:
      Murder Assault UrbanPop      Rape
1 1.0000000 0.80000000      0 0.40000000
2 0.0000000 0.04347826      0 0.08695652
3 0.7142857 1.00000000      1 0.85714286
4 0.1000000 0.00000000      1 0.20000000

Clustering vector:
           Alabama      Alaska      Arizona      Arkansas      California
1              1          1          3          2                  3
           Colorado   Connecticut   Delaware      Florida      Georgia
4              4          4          2          3                  1
           Hawaii      Idaho      Illinois      Indiana      Iowa
4              2          2          3          2                  2
           Kansas      Kentucky   Louisiana      Maine      Maryland
2              2          2          1          2                  1
           Massachusetts Michigan   Minnesota      Mississippi      Missouri
4              3          3          2          1                  2
           Montana      Nebraska   Nevada   New Hampshire      New Jersey
2              2          2          3          2                  4
           New Mexico   New York   North Carolina   North Dakota      Ohio
1              3          3          1          2                  4
           Oklahoma      Oregon   Pennsylvania   Rhode Island   South Carolina
2              2          2          2          4                  1
           South Dakota Tennessee   Texas      Utah      Vermont
2              1          1          4          4                  2
           Virginia      Washington West Virginia   Wisconsin      Wyoming
2              4          4          2          2                  2

Within cluster sum of squares by cluster:
[1] 4.000000 2.782609 2.285714 2.500000
  (between_SS / total_SS =  73.1 %)

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

```



Result:

Thus, the program has been executed and the output has been verified.