SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

PRACTICAL RECORD NOTE

STUDENT NAME	:		
REGISTER NUMBER	:		
CLASS	: MCA	**SECTION**	: G
YEAR & SEMESTER	: I YEAR & II SEM		
SUBJECT CODE	: PCA20S02J		
SUBJECT TITLE	: DATA ANALYSIS USING R		

APRIL 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

SRM Nagar, Kattankulathur – 603 203

CERTIFICATE

Certified to be the bonafide record of practical work done by

____

*Register No.*___________________________*of* _____*MCA*___________*Degree*

course for PCA20S02J – DATA ANALYSIS USING R in the Computer lab in SRM Institute of Science and Technology during the academic year2023-2024.

Staff In-charge *Head of the Department*

*Submitted for Semester Practical Examination held on*____________________.

Internal Examiner *External Examiner*

INDEX

EX.NO : 01

DATE :

INSTALLATION OF R AND R-STUDIO

AIM :

To install R and R-Studio.

PROCEDURE :

INSTALLATION OF R

STEP 1 : Go to CRAN R project website - https://cran.r-project.org/

STEP 2 : Click on the Download R for Windows link.

STEP 3 : Click on the base subdirectory link or install R for the first time link.

STEP 4 : Click Download R 4.2.1 for Windows and save the executable .exe file.

STEP 5 : Run the .exe file and follow the installation instructions.

STEP 6 : Select the desired language and then click Next.

STEP 7 : Read the license agreement and click Next.

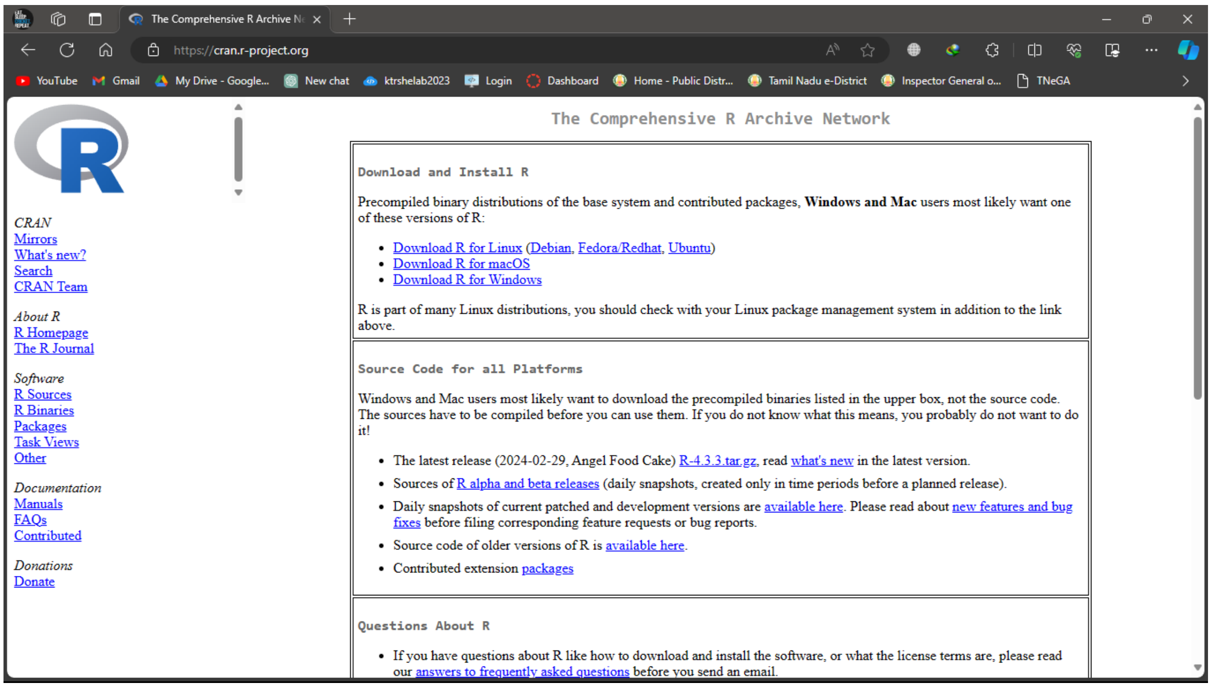
STEP 8 : Select the components you wish to install, Click Next.

STEP 9 : Browse the folder/path you wish to install R into and then confirm by clicking Next.

STEP 10 : Select additional tasks like creating desktop shortcuts etc, then click Next.

STEP 11 : Wait for the installation process to complete.

OUTPUT :



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Donations
Donate

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

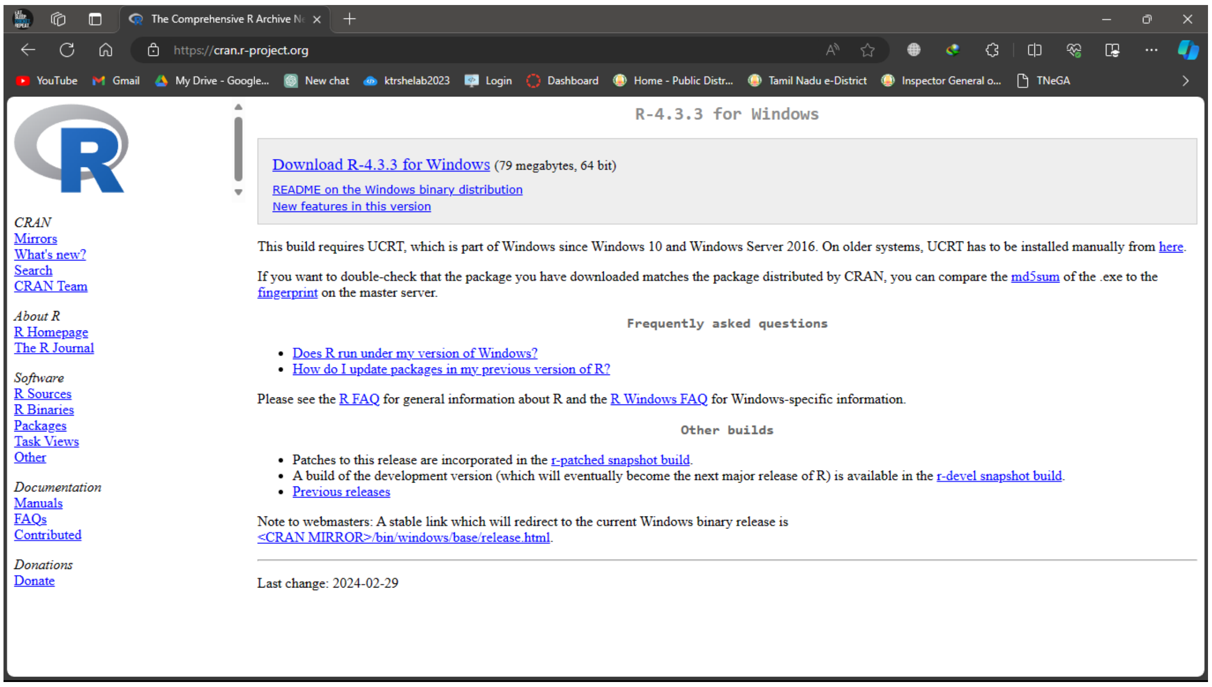
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2024-02-29, Angel Food Cake) R-4.3.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Donations
Donate

R-4.3.3 for Windows

Download R-4.3.3 for Windows (79 megabytes, 64 bit)
README on the Windows binary distribution
New features in this version

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from here.

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 fingerprint on the master server.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2024-02-29

INSTALLATION OF R – STUDIO

STEP 1 : With R-base installed, let's move on to installing R-Studio. To begin, go to download RStudio - https://posit.co/ downloads and click on the download button for R-Studio desktop.

STEP 2 : Click on the link for the windows version of R-Studio and save the .exe file.

STEP 3 : Run the .exe and follow the installation instructions.

STEP 4 : Click Next on the welcome window.

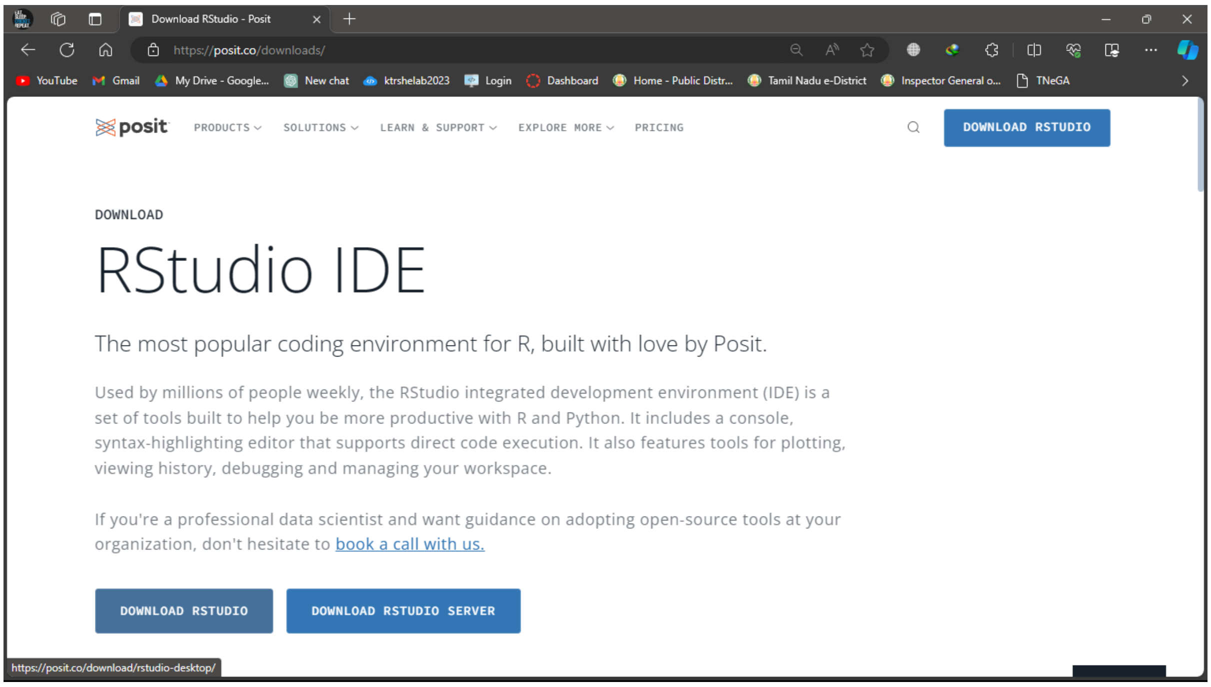
STEP 5 : Enter/browse the path to the installation folder and click Next to proceed.

STEP 6 : Select the folder for the start menu shortcut and then click Next.

STEP 7 : Wait for the installation process to complete.

STEP 8 : Click on Finish to complete the installation.

OUTPUT :



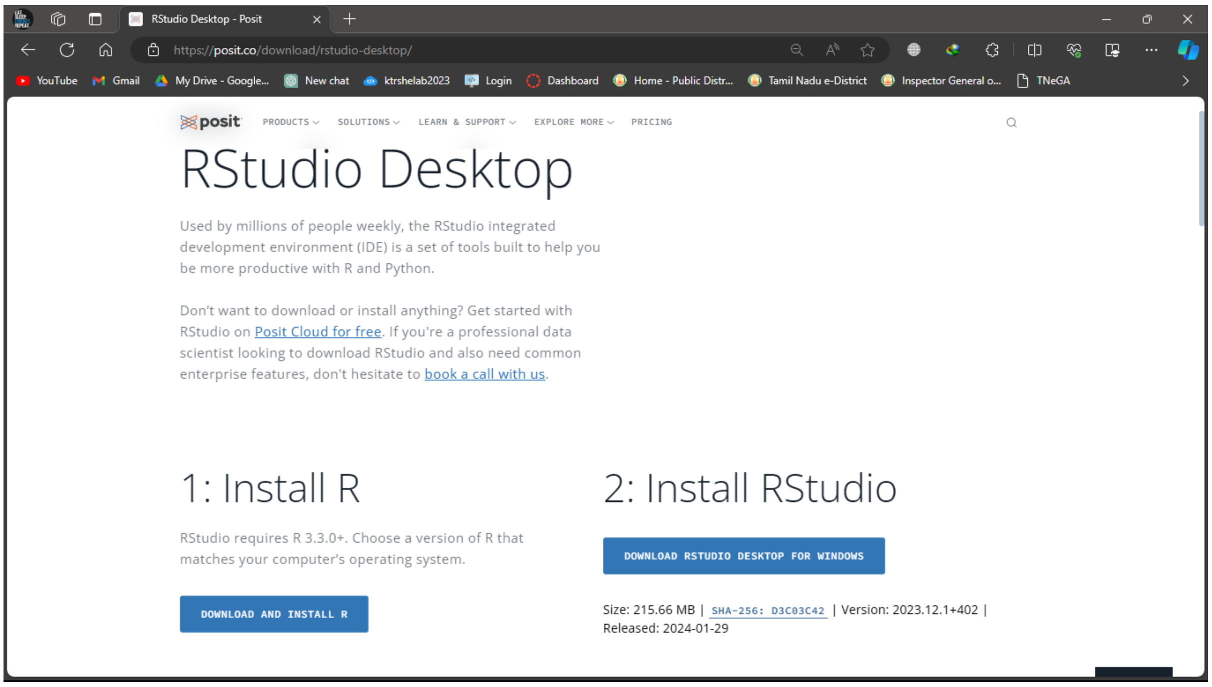
DOWNLOAD

RStudio IDE

The most popular coding environment for R, built with love by Posit.

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution. It also features tools for plotting, viewing history, debugging and managing your workspace.

If you're a professional data scientist and want guidance on adopting open-source tools at your organization, don't hesitate to book a call with us.



RStudio Desktop

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.

Don't want to download or install anything? Get started with RStudio on Posit Cloud for free. If you're a professional data scientist looking to download RStudio and also need common enterprise features, don't hesitate to book a call with us.

1: Install R

RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

DOWNLOAD AND INSTALL R

2: Install RStudio

DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS

Size: 215.66 MB | SHA-256: D3C03C42 | Version: 2023.12.1+402 | Released: 2024-01-29

EX.NO : 02

DATE :

WORKING WITH R PACKAGES

AIM :

To install, load and work with packages.

PACKAGES :

INSTALL R PACKAGES FROM CRAN :

install.packages ( ) - This function is used to install a required package in the R programming language.

Example :- To install ggplot2 package

install.packages("ggplot2")

UNINSTALL R PACKAGES :

remove.packages ( ) - This function is used to uninstall apackage in the R programming language.

Example :- To uninstall ggplot2 package

remove.packages("ggplot2")

LOADING OF R PACKAGES :

library ( ) - It is used to load and list all the packages in the R Programming language.

Example :- To load ggplot2 package

library (ggplot2)

To list all the packages installed

library ( )

UPDATING R PACKAGES :

old.packages ( ) - It is used to check which packages need an update in R.

Example :- To check an update

old.packages( )

update.packages ( ) - It is used to update all the packages in R .

Example :- To update Packages

update.packages( )

LISTING THE PACKAGES THAT ARE INSTALLED :

install.packages ( ) - It is used to list out all packages which are installed in computer .

Example :- To list out installed packages

installed.packages( )

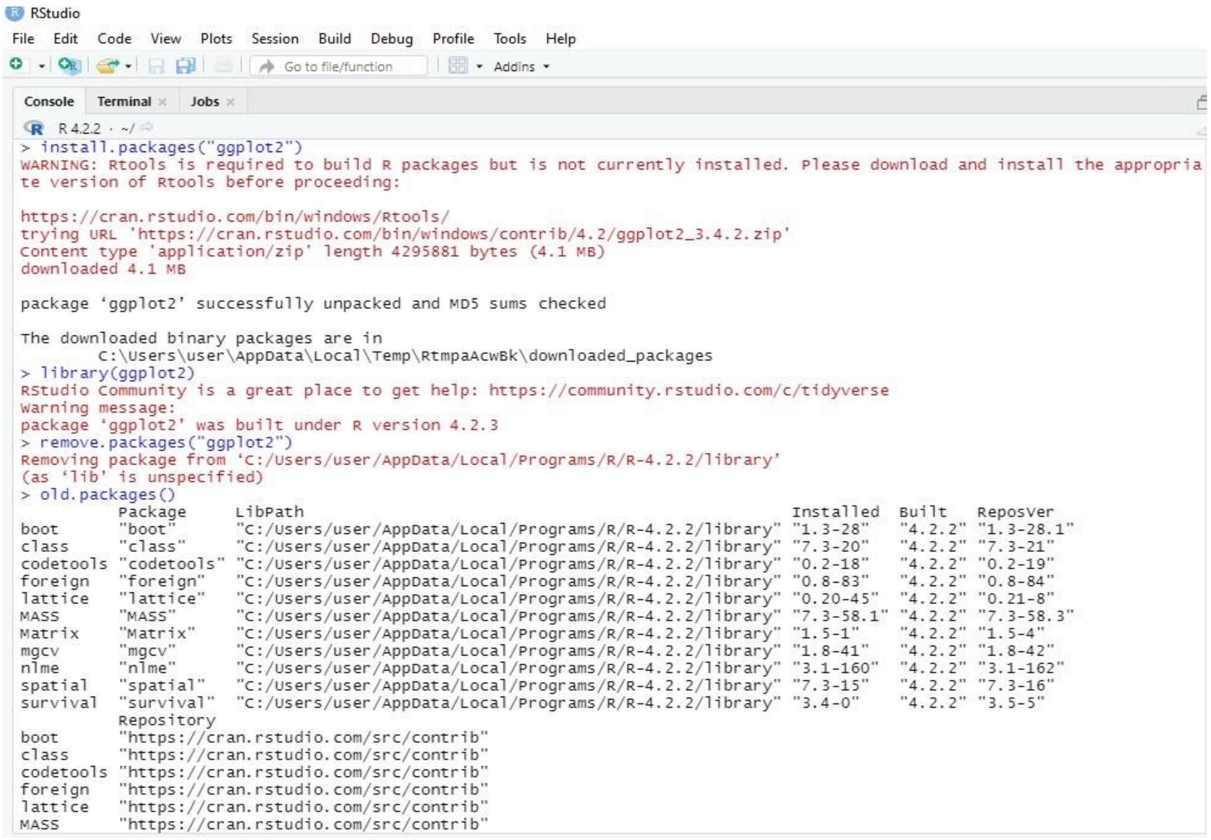
GET HELP PAGES ABOUT PACKAGES :

Help ( ) and (?) - They provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages.

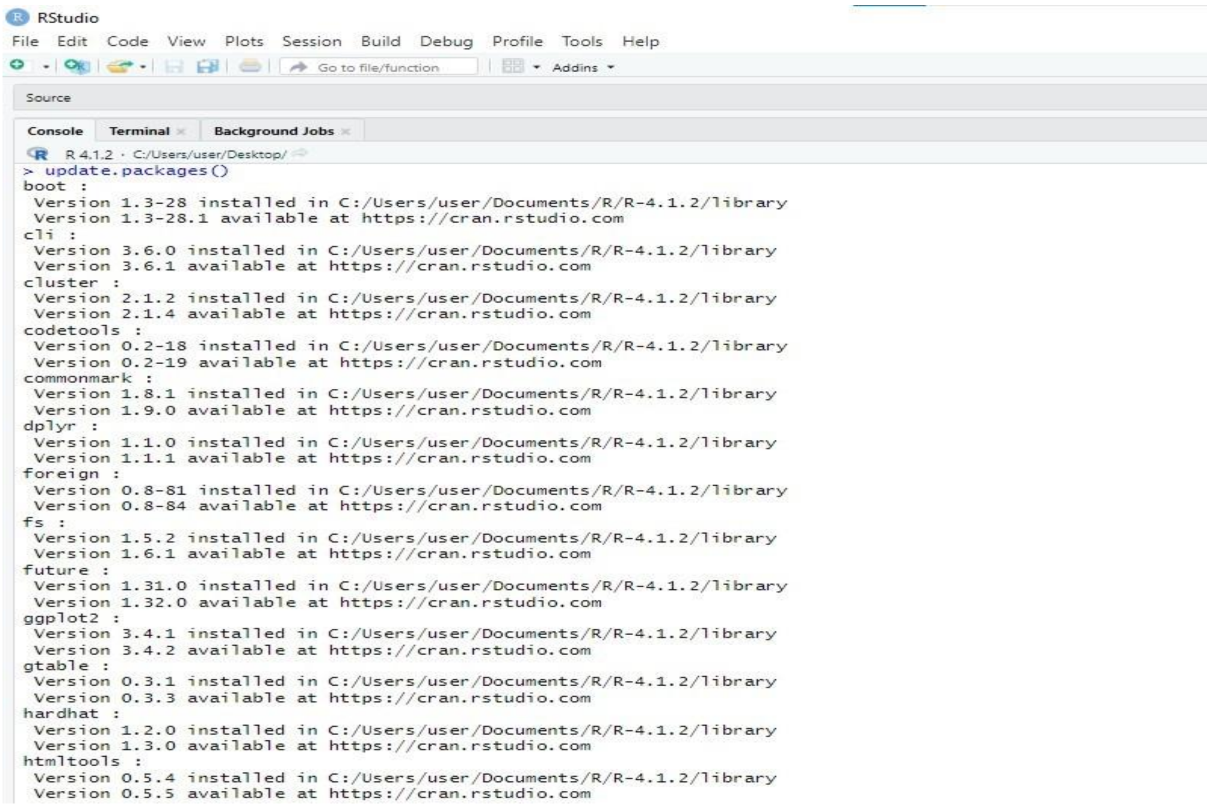
Example :- To get more description about ggplot2

help("ggplot2") and ?ggplot2

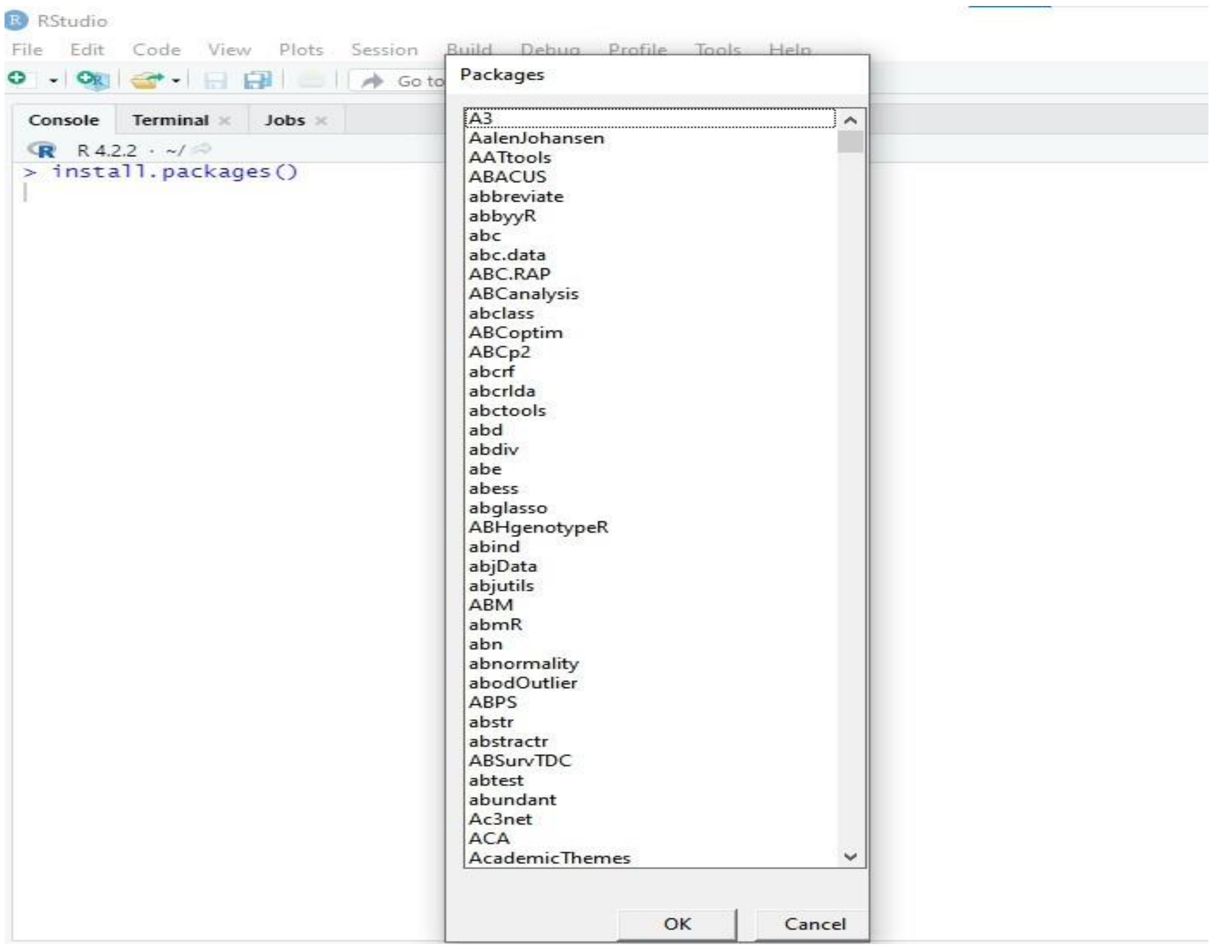
OUTPUT :


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Console
Terminal
Jobs
R 4.2.2 · ~/

```
> install.packages("ggplot2")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropria
te version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/ggplot2_3.4.2.zip'
Content type 'application/zip' length 4295881 bytes (4.1 MB)
downloaded 4.1 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpaAcwBk\downloaded_packages
> library(ggplot2)
RStudio Community is a great place to get help: https://community.rstudio.com/c/tidyverse
Warning message:
package 'ggplot2' was built under R version 4.2.3
> remove.packages("ggplot2")
Removing package from 'C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library'
(as 'lib' is unspecified)
> old.packages()
          Package     LibPath                                                Installed  Built   ReposVer
boot      "boot"      "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "1.3-28"   "4.2.2" "1.3-28.1"
class     "class"     "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "7.3-20"   "4.2.2" "7.3-21"
codetools "codetools" "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "0.2-18"   "4.2.2" "0.2-19"
foreign   "foreign"   "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "0.8-83"   "4.2.2" "0.8-84"
lattice   "lattice"   "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "0.20-45"  "4.2.2" "0.21-8"
MASS      "MASS"      "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "7.3-58.1" "4.2.2" "7.3-58.3"
Matrix    "Matrix"    "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "1.5-1"    "4.2.2" "1.5-4"
mgcv      "mgcv"      "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "1.8-41"   "4.2.2" "1.8-42"
nlme      "nlme"      "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "3.1-160"  "4.2.2" "3.1-162"
spatial   "spatial"   "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "7.3-15"   "4.2.2" "7.3-16"
survival  "survival"  "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library" "3.4-0"    "4.2.2" "3.5-5"
          Repository
boot      "https://cran.rstudio.com/src/contrib"
class     "https://cran.rstudio.com/src/contrib"
codetools "https://cran.rstudio.com/src/contrib"
foreign   "https://cran.rstudio.com/src/contrib"
lattice   "https://cran.rstudio.com/src/contrib"
MASS      "https://cran.rstudio.com/src/contrib"
```

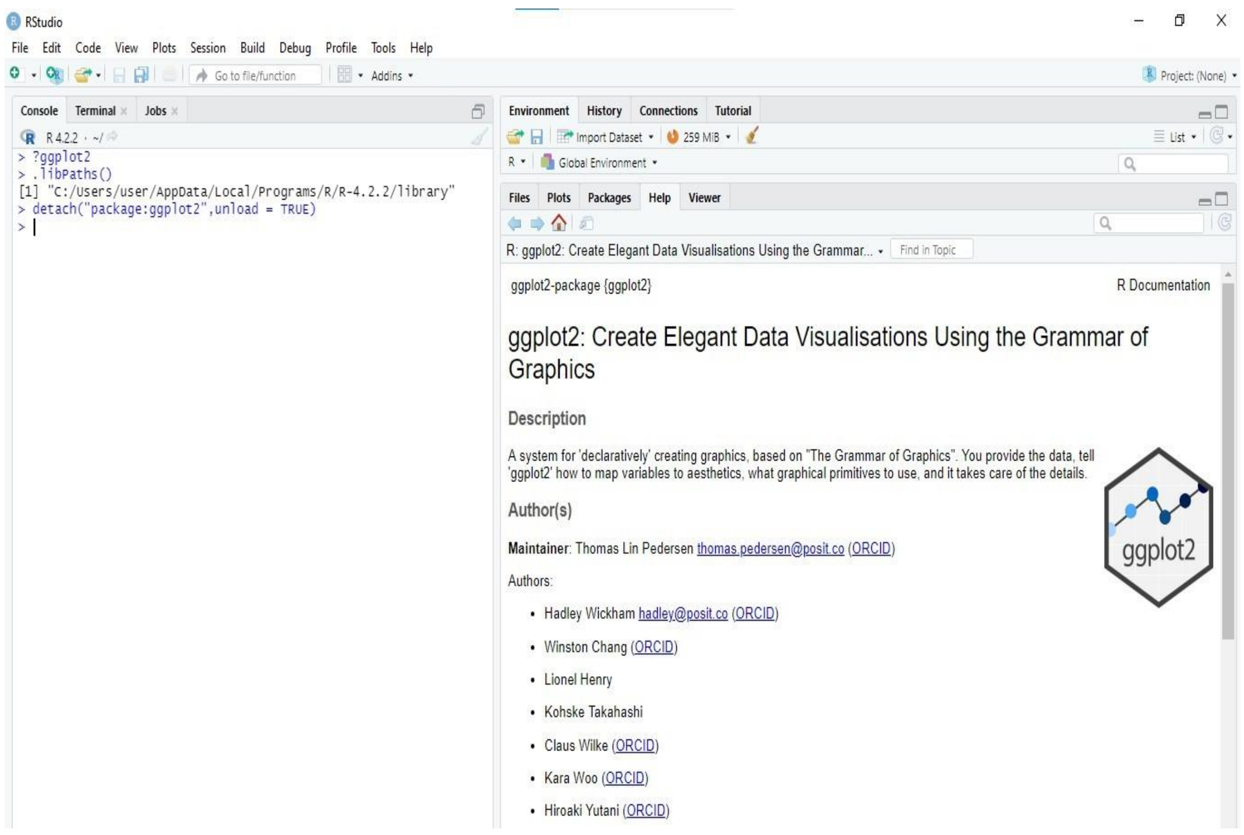

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.1.2 · C:/Users/user/Desktop/

```
> update.packages()
boot :
 Version 1.3-28 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 1.3-28.1 available at https://cran.rstudio.com
cli :
 Version 3.6.0 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 3.6.1 available at https://cran.rstudio.com
cluster :
 Version 2.1.2 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 2.1.4 available at https://cran.rstudio.com
codetools :
 Version 0.2-18 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 0.2-19 available at https://cran.rstudio.com
commonmark :
 Version 1.8.1 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 1.9.0 available at https://cran.rstudio.com
dplyr :
 Version 1.1.0 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 1.1.1 available at https://cran.rstudio.com
foreign :
 Version 0.8-81 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 0.8-84 available at https://cran.rstudio.com
fs :
 Version 1.5.2 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 1.6.1 available at https://cran.rstudio.com
future :
 Version 1.31.0 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 1.32.0 available at https://cran.rstudio.com
ggplot2 :
 Version 3.4.1 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 3.4.2 available at https://cran.rstudio.com
gtable :
 Version 0.3.1 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 0.3.3 available at https://cran.rstudio.com
hardhat :
 Version 1.2.0 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 1.3.0 available at https://cran.rstudio.com
htmltools :
 Version 0.5.4 installed in C:/Users/user/Documents/R/R-4.1.2/library
 Version 0.5.5 available at https://cran.rstudio.com
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to
Console
Terminal
Jobs
R 4.2.2 · ~/
Packages
OK
Cancel

```
> install.packages()
|
```

A3
AalenJohansen
AATtools
ABACUS
abbreviate
abbyyR
abc
abc.data
ABC.RAP
ABCanalysis
abclass
ABCoptim
ABCp2
abcrf
abcrlda
abctools
abd
abdiv
abe
abess
abglasso
ABHgenotypeR
abind
abjData
abjutils
ABM
abmR
abn
abnormality
abodOutlier
ABPS
abstr
abstractr
ABSurvTDC
abtest
abundant
Ac3net
ACA
AcademicThemes



```
> ?ggplot2
> .libPaths()
[1] "C:/Users/user/AppData/Local/Programs/R/R-4.2.2/library"
> detach("package:ggplot2",unload = TRUE)
> |
```

ggplot2-package {ggplot2} R Documentation

ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics

Description

A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Author(s)

Maintainer: Thomas Lin Pedersen thomas.pedersen@posit.co (ORCID)

Authors:

- Hadley Wickham hadley@posit.co (ORCID)
- Winston Chang (ORCID)
- Lionel Henry
- Kohske Takahashi
- Claus Wilke (ORCID)
- Kara Woo (ORCID)
- Hiroaki Yutani (ORCID)

EX.NO : 03

DATE :

BUILT – IN FUNCTIONS

AIM :

To show the working of Built – In Functions.

BUILT – IN FUNCTION’S :

FUNCTIONS	DESCRIPTIONS
sum ( )	It returns the sum of all the input vector.
prod ( )	It returns the multiplication result of all the input vector.
max ( )	It returns the maximum value of input vector.
min ( )	It returns the minimum value of input vector.
unique ( )	It returns the unique value.
sort ( )	It returns sorted value from input vector.
rev ( )	It returns reversed order of input vector.
rbind ( )	It combines vector and matrix row-wise.
cbind ( )	It combines vector and matrix column-wise.
setdiff ( )	It returns differences between two vectors.
cumsum ( )	It returns the cumulative sum of two vector.

abs ( )	It returns the absolute value of input vector.
sqrt ( )	It returns the square root of input vector.
ceiling ( )	It returns the smallest integer which is larger than or equal to input vector.
floor ( )	It returns the largest integer, which is smaller than or equal to input vector.
trunc ( )	It returns the truncate value of input vector.
round ( )	It returns round value of input vector.
cos( ), sin(), tan( )	It returns cos( ), sin( ), tan( ) value of input vector.
log ( )	It returns natural logarithm of input vector.
log10 ( )	It returns common logarithm of input vector.
exp ( )	It returns exponent of input vector.

PROGRAM :

1) Create a Vector ‘v’ with the values 1,5,8,10,4,5,3,9,8,10,12 and do the following :-

a) Find Sum, Mean and Product of the vector

```
v =c(1,5,8,10,4,5,3,9,8,10,12)

sum(v)      mean(v)    prod(v)
```

b) Find the Minimum and the Maximum of the vector.

```
min(v)      max(v)
```

c) Sort the vector in Ascending and Descending order.

sort(v)   sort(v,decreasing=TRUE)

d) List the Distinct values in the vector.

unique(v)

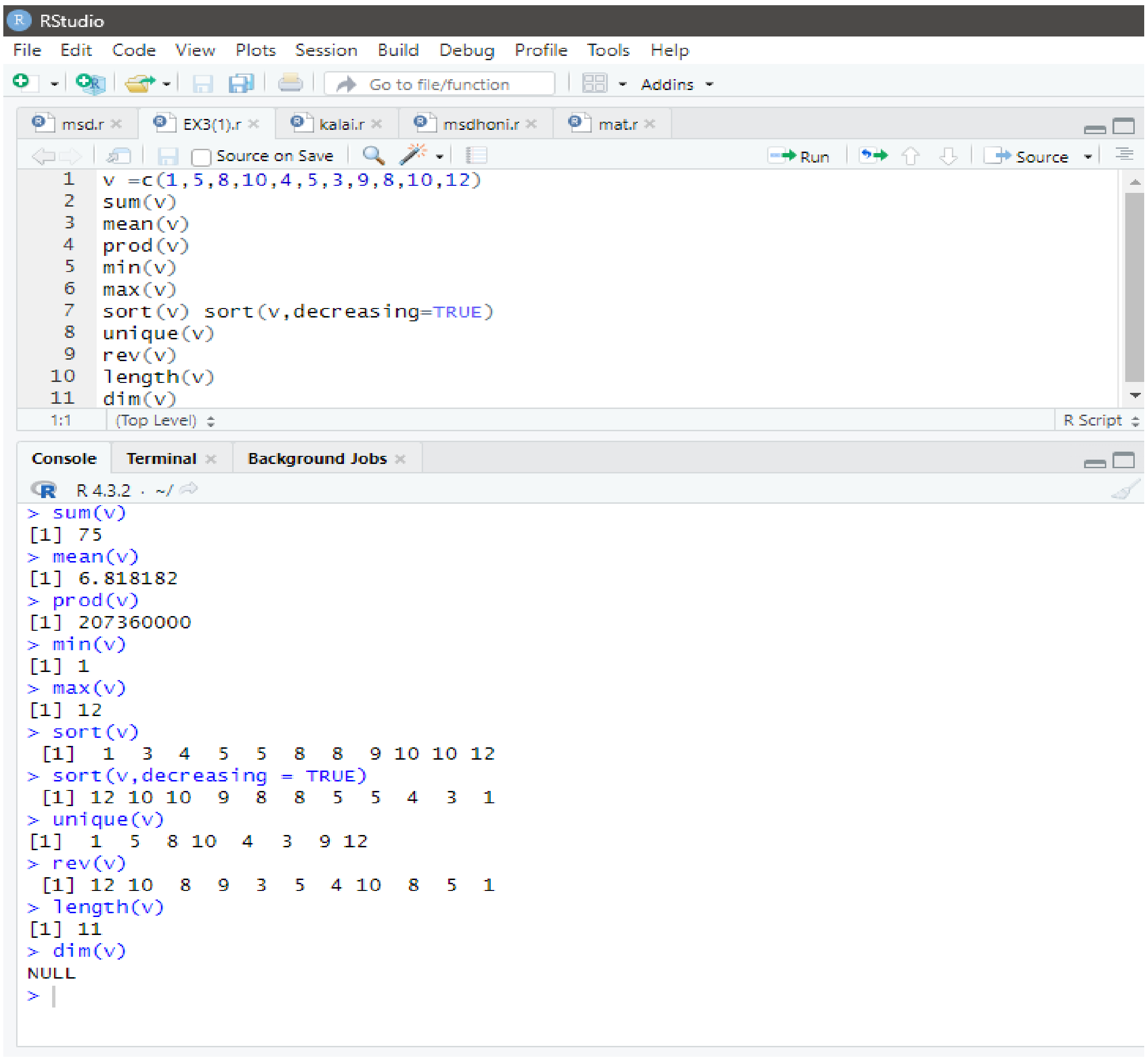
e) Reverse the order of the vector.

rev(v)

f) Find the Length and the Dimension of the vector.

length(v)   dim(v)

OUTPUT :



```
v =c(1,5,8,10,4,5,3,9,8,10,12)
sum(v)
mean(v)
prod(v)
min(v)
max(v)
sort(v) sort(v,decreasing=TRUE)
unique(v)
rev(v)
length(v)
dim(v)
```

```
> sum(v)
[1] 75
> mean(v)
[1] 6.818182
> prod(v)
[1] 207360000
> min(v)
[1] 1
> max(v)
[1] 12
> sort(v)
 [1]  1  3  4  5  5  8  8  9 10 10 12
> sort(v,decreasing = TRUE)
 [1] 12 10 10  9  8  8  5  5  4  3  1
> unique(v)
[1]  1  5  8 10  4  3  9 12
> rev(v)
 [1] 12 10  8  9  3  5  4 10  8  5  1
> length(v)
[1] 11
> dim(v)
NULL
> |
```

2) Create two vectors as below :-

A = 0,2,4,15

B = 3,12,4,11

a) Combines these two vectors by column wise and row wise.

A = c(0,2,4,15)

B = c(3,12,4,11)

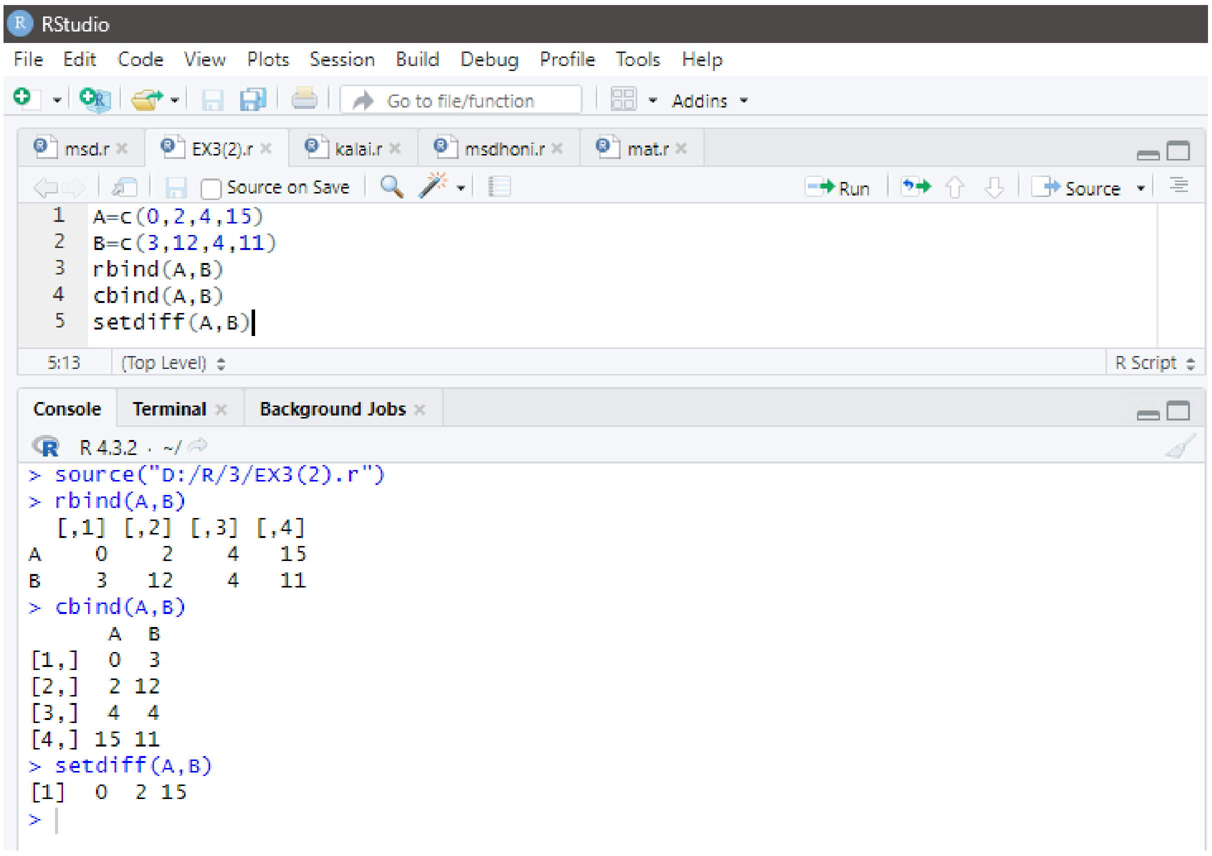
rbind(A,B)

cbind(A,B)

b) Find the elements of a ‘A’ vector that are not in ‘B’ vector.

setdiff(A,B)

OUTPUT :



```
A=c(0,2,4,15)
B=c(3,12,4,11)
rbind(A,B)
cbind(A,B)
setdiff(A,B)
```

```
> source("D:/R/3/EX3(2).r")
> rbind(A,B)
  [,1] [,2] [,3] [,4]
A    0    2    4   15
B    3   12    4   11
> cbind(A,B)
      A  B
[1,]  0  3
[2,]  2 12
[3,]  4  4
[4,] 15 11
> setdiff(A,B)
[1]  0  2 15
> 
```

3) A survey asks people if they smoke or not.The data is yes, no, no, yes, yes.

a) Represent the above information in a vector.

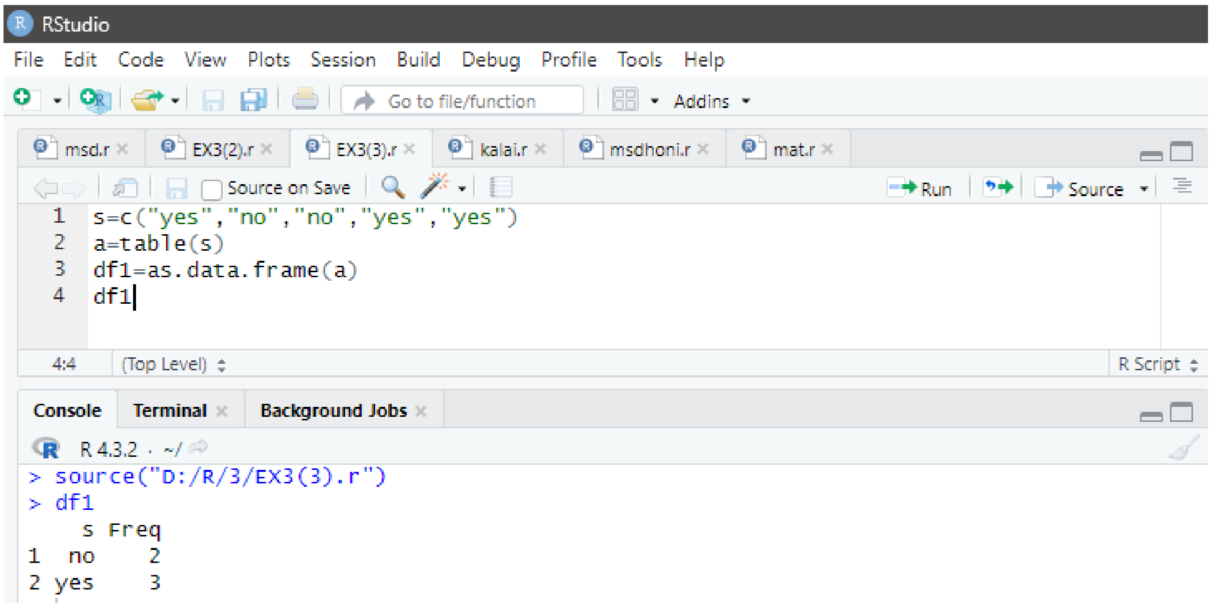
s=c("yes","no","no","yes","yes")

b) Display frequency table for above information.

a=table(s)

df1=as.data.frame(a)   df1

OUTPUT :



```
s=c("yes","no","no","yes","yes")
a=table(s)
df1=as.data.frame(a)
df1
```

```
> source("D:/R/3/EX3(3).r")
> df1
    s Freq
1  no    2
2 yes    3
```

EX.NO : 04

DATE :

USER - DEFINED FUNCTIONS

AIM :

To create and use User Defined function.

USER DEFINED FUNCTION'S :

To declare a user-defined function in R, we use the keyword function.

Syntax :

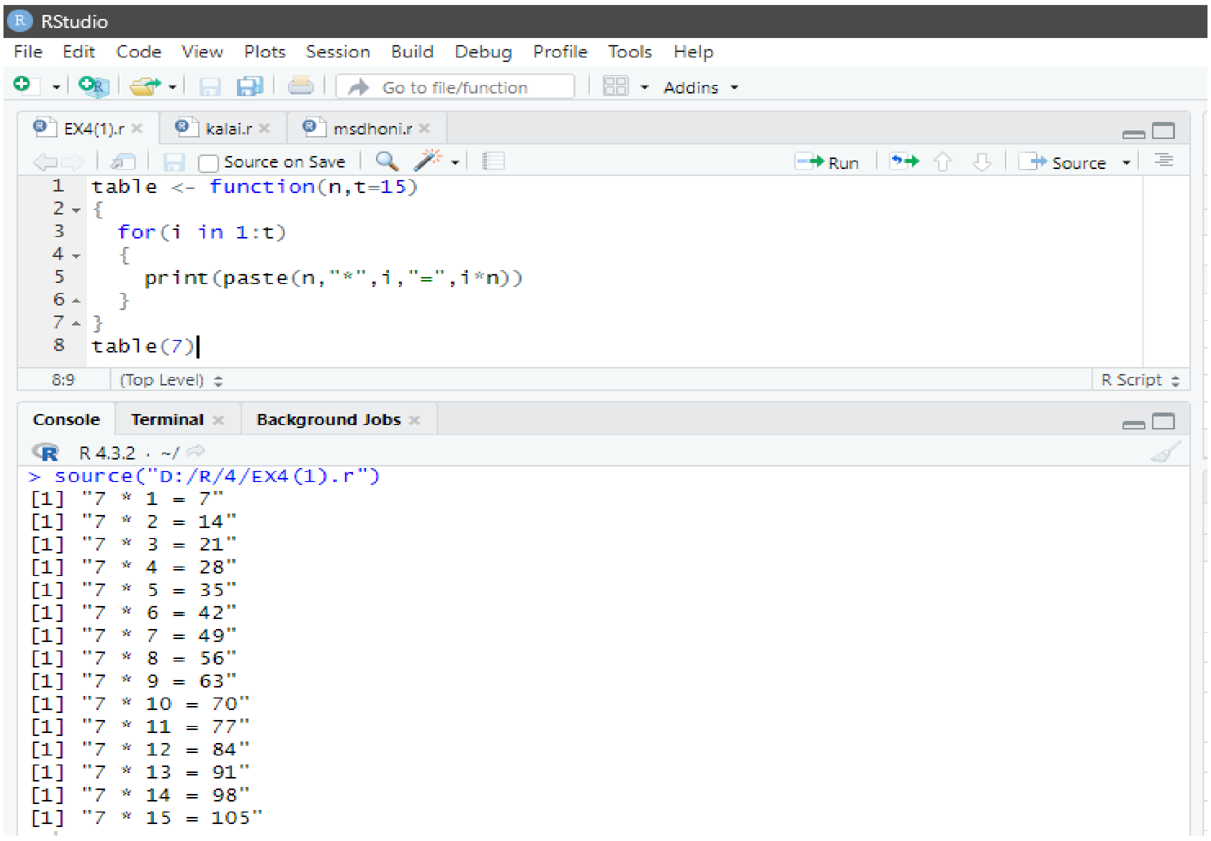
```
function_name <- function(parameters)
{
        function body
}
```

PROGRAM :

1) Write a function "tables" in R with argument for table number and times to display the respective multiplication table. Use default value 15 for times.

```
table <- function(n,t=15)
{
    for(i in 1:t)
    {
        print(paste(n,"*",i,"=",i*n))
    }
}
table(7)
```

OUTPUT :



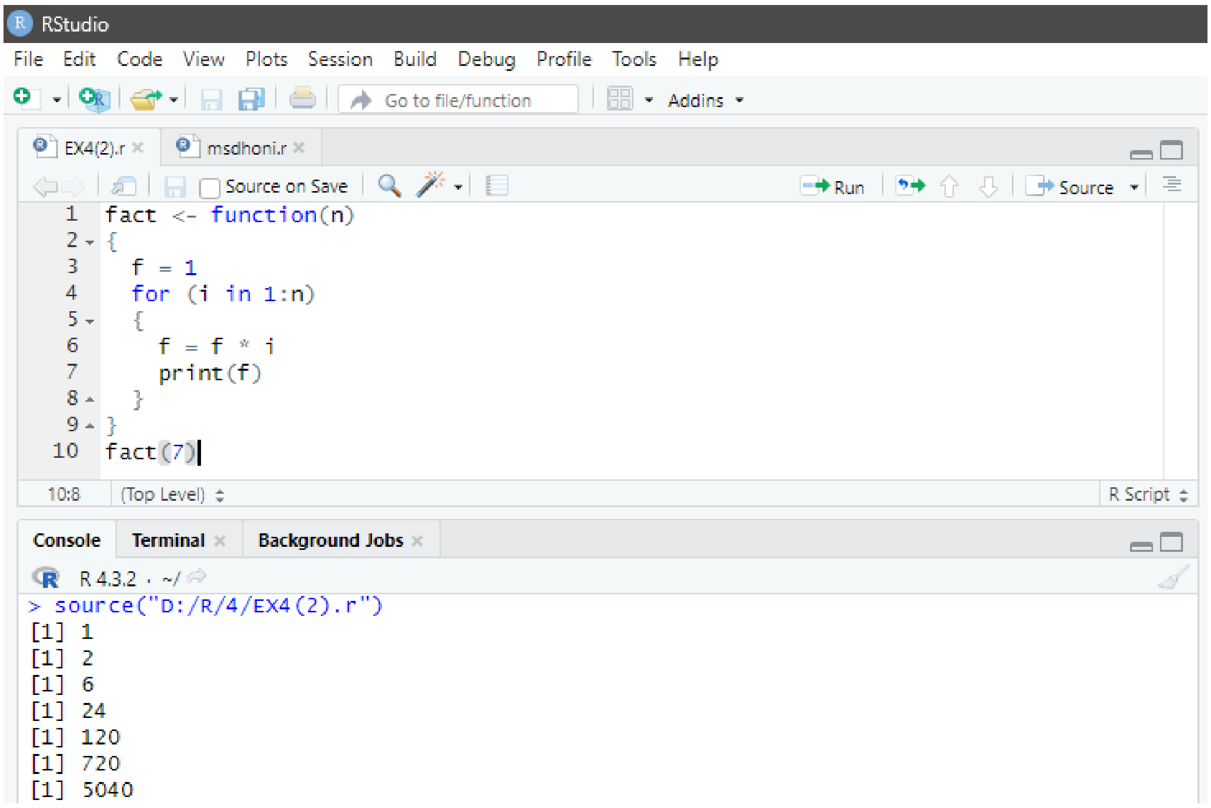
```
table <- function(n,t=15)
{
  for(i in 1:t)
  {
    print(paste(n,"*",i,"=",i*n))
  }
}
table(7)
```

```
> source("D:/R/4/EX4(1).r")
[1] "7 * 1 = 7"
[1] "7 * 2 = 14"
[1] "7 * 3 = 21"
[1] "7 * 4 = 28"
[1] "7 * 5 = 35"
[1] "7 * 6 = 42"
[1] "7 * 7 = 49"
[1] "7 * 8 = 56"
[1] "7 * 9 = 63"
[1] "7 * 10 = 70"
[1] "7 * 11 = 77"
[1] "7 * 12 = 84"
[1] "7 * 13 = 91"
[1] "7 * 14 = 98"
[1] "7 * 15 = 105"
```

2) Write a function ‘fact’ in R to display the factorial value of the passed argument.

```
fact <- function(n)
{
    f = 1
    for (i in 1:n)
    {
            f = f * i
            print(f)
    }
}
fact(7)
```

OUTPUT :



```
fact <- function(n)
{
  f = 1
  for (i in 1:n)
  {
    f = f * i
    print(f)
  }
}
fact(7)
```

```
> source("D:/R/4/EX4(2).r")
[1] 1
[1] 2
[1] 6
[1] 24
[1] 120
[1] 720
[1] 5040
```

EX.NO : 05

DATE :

DESCRIPTIVE STATISTICS

AIM :

To explore the commands that will give an overview of data to be used.

DESCRIPTIVE STATISTICS:

Descriptive statistics is the branch of statistics that focuses on describing and gaining more insight into the data in its present state.

FUNCTIONS	DESCRIPTIONS
mean()	It returns arithmetic average of a numeric input vector.
median()	It returns median of a numeric input vector.
var()	It returns variance of a numeric input vector.
sd()	It returns standard deviation of a numeric input vector.
range()	It returns the maximum and minimum value of a numeric input vector.
diff()	It returns the difference between pairs of consecutive elements of a numeric vector.
IQR()	It returns the interquartile range of a numeric input vector.
quantile()	It returns the sample quantile of a numeric input vector.
summary()	It returns summary statistics such as mean, median, minimum, maximum, 1st quantile, 3rd quantile, etc. for each component in the object.

str()	It displays the internal structure of an R object.
table()	It performs a tabulation of categorical variable and gives its frequency as output.
prop.table()	It calculates the proportions of a table, with the result presented as a table with proportions.

PROGRAM :

1) Display the structure and summary statistics of the iris dataset.

```
str(iris) summary(iris)
```

2) Find the mean of Sepal.Length of iris dataset.

```
mean(iris$Sepal.Length)
```

3) Display variance of Sepal.Length of iris dataset.

```
var(iris$Sepal.Length)
```

4) Display median of Petal.Length of iris dataset.

```
median(iris$Petal.Length)
```

5) Find Standard Deviation of Setal.Length of iris dataset.

```
sd(iris$Sepal.Length)
```

6) Display top 10 records from the dataset iris.

```
head(iris, n = 10)
```

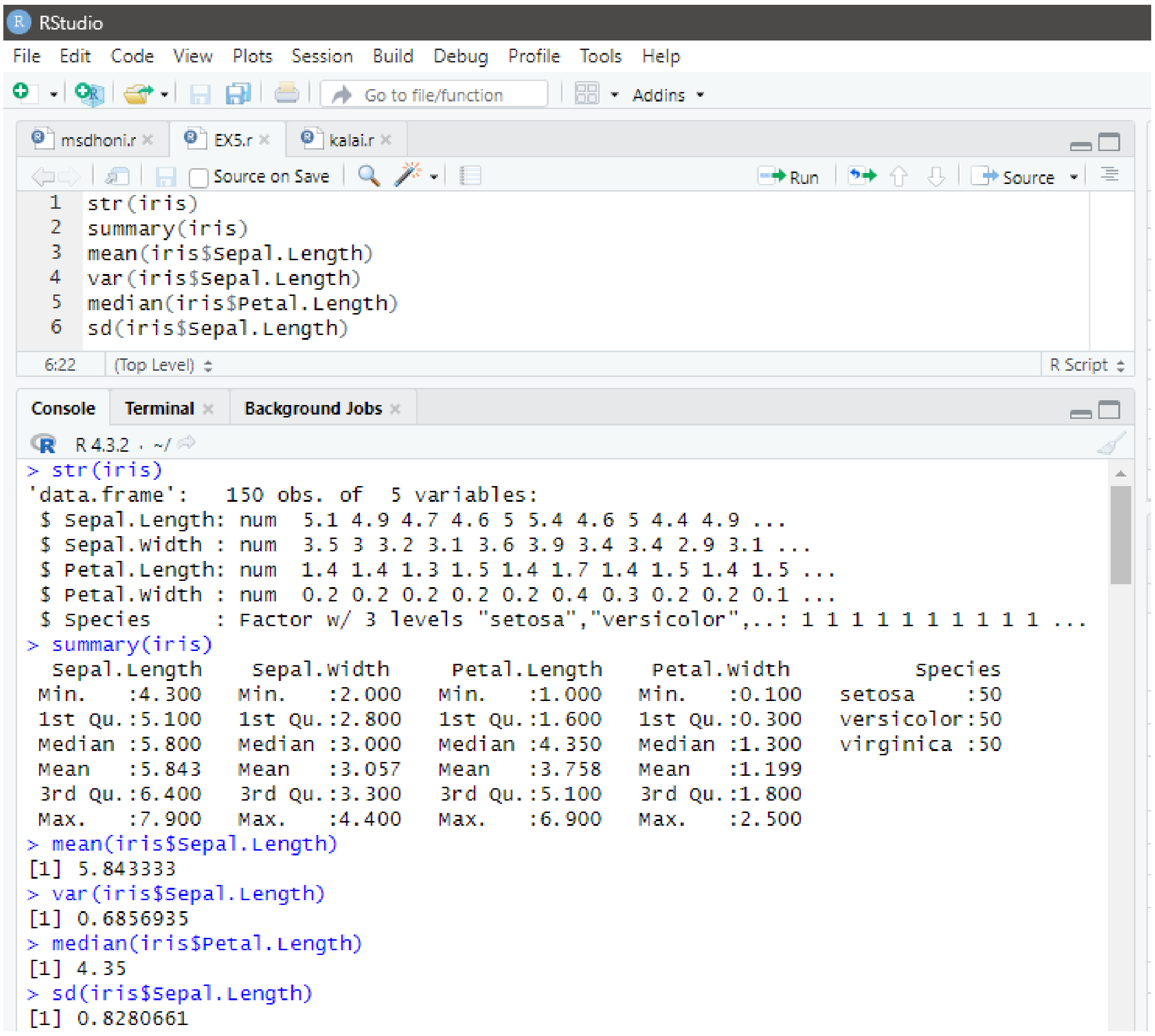
7) Display bottom 10 records from the dataset iris.

```
tail(iris, n=10)
```

8) Print contents of the dataframe iris.

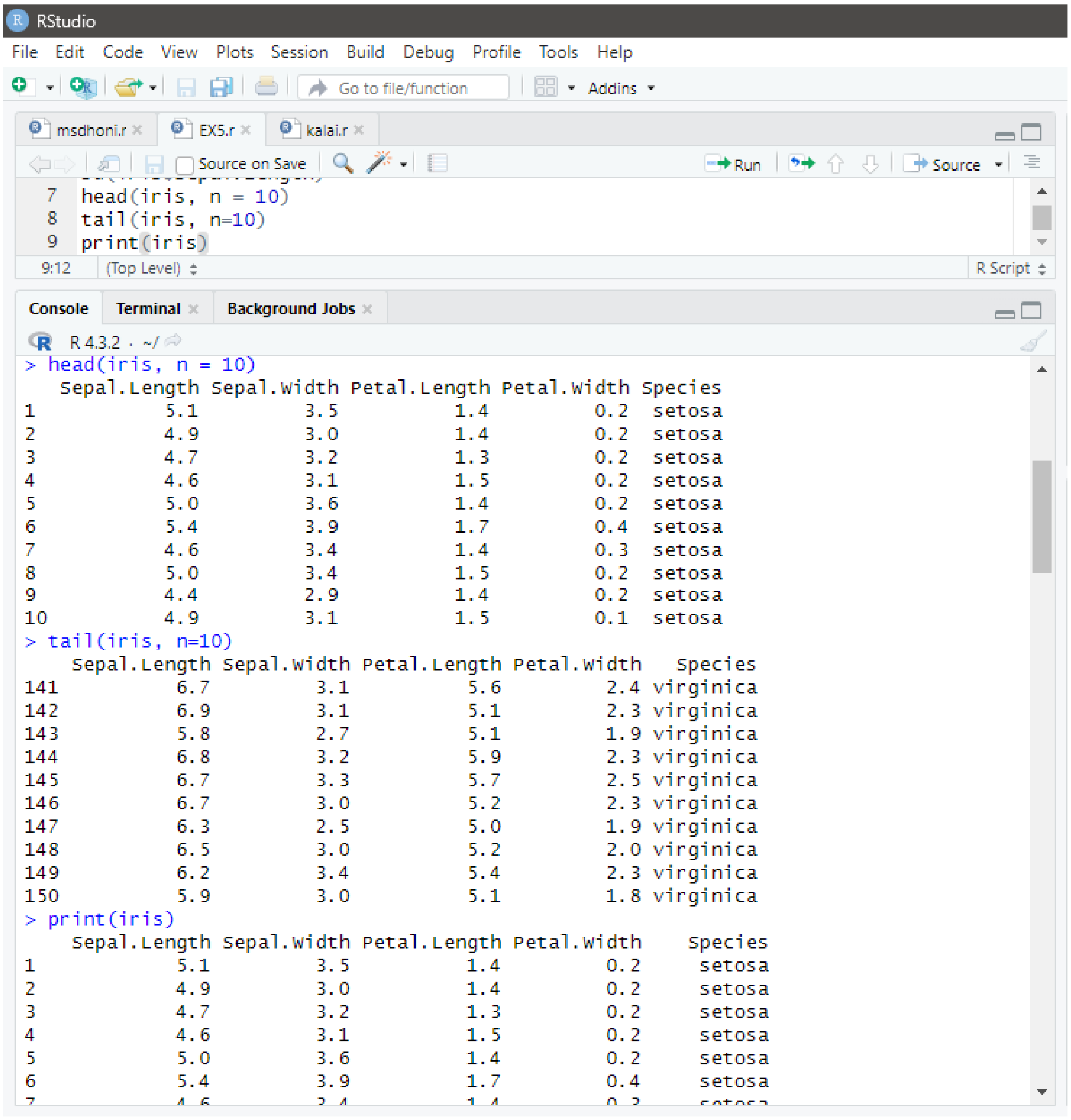
```
print(iris)
```

OUTPUT :



```
str(iris)
summary(iris)
mean(iris$Sepal.Length)
var(iris$Sepal.Length)
median(iris$Petal.Length)
sd(iris$Sepal.Length)
```

```
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> 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
> mean(iris$Sepal.Length)
[1] 5.843333
> var(iris$Sepal.Length)
[1] 0.6856935
> median(iris$Petal.Length)
[1] 4.35
> sd(iris$Sepal.Length)
[1] 0.8280661
```



```
head(iris, n = 10)
tail(iris, n=10)
print(iris)
```

```
> head(iris, n = 10)
   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
> tail(iris, n=10)
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
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
> print(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
```

EX.NO : 06

DATE :

VECTOR MANIPULATION

AIM :

To create and manipulate vector in R.

VECTOR :

A vector is asequence of data elements of the same basic type. Data types can be numeric, integer,character, complex or logical. It is created using the c() function. Since, a vector must have elements of thesame type, this function will try and coerce elements to the same type, if they are different.Coercion is from lower to higher types from logical to integer to double to character.

PROGRAM :

1) Create the following vectors.

Note : use : operator, seq() and rep()

a) (1, 2, 3, . . . , 19, 20).

seq(1,20)

b) (20, 19, . . . , 2, 1).

seq(20,1)

c) (1, 2, 3, . . . , 19, 20, 19, 18, . . . , 2, 1).

c(1:20,19:)

d) (5,10,15, … , 100).

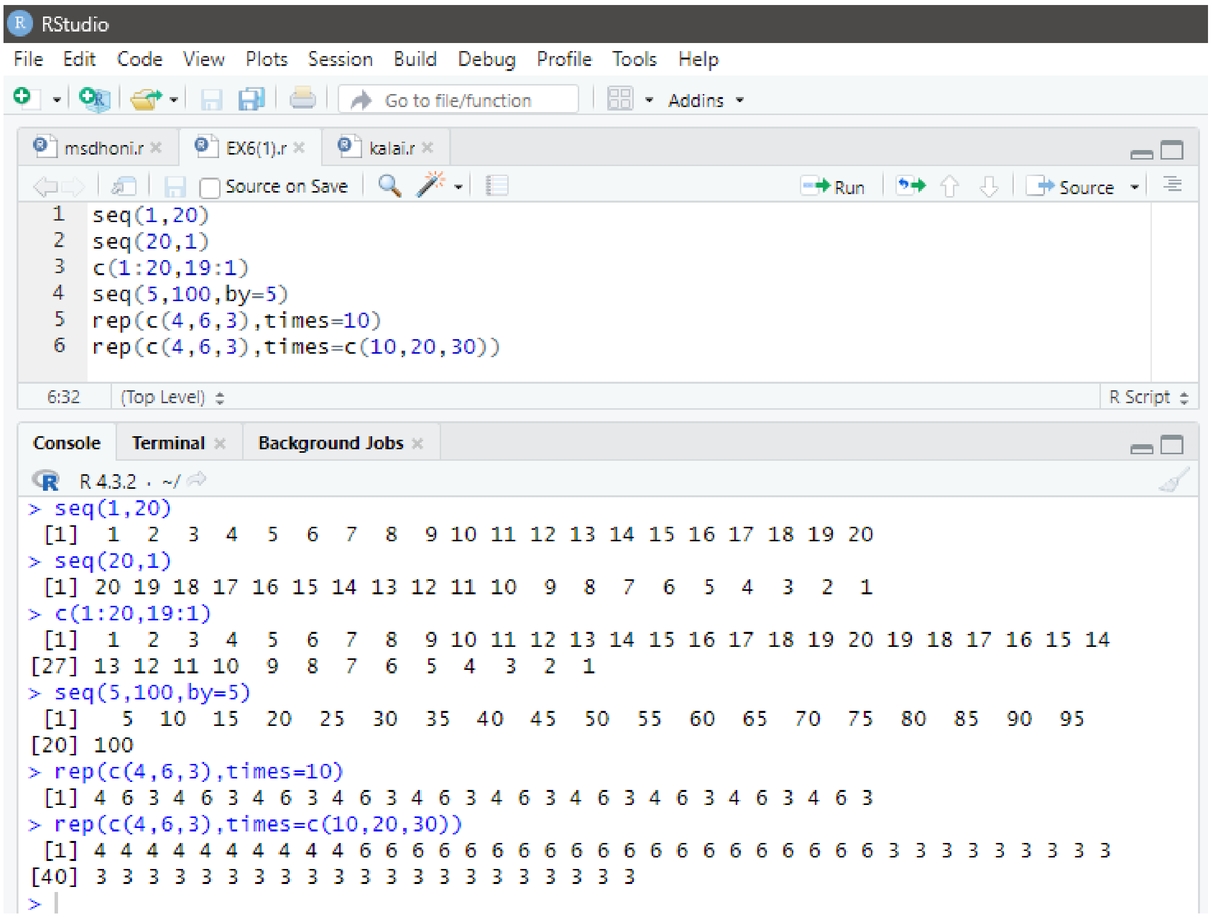
seq(5,100,by=5)

e) (4, 6, 3, 4, 6, 3, . . . , 4, 6, 3) where there are 10 occurrences of 4,6,3. **rep(c(4,6,3),times=10)**

f) (4, 4, . . . , 4, 6, 6, . . . , 6, 3, 3, . . . , 3) where there are 10 occurrences of 4, 20 occurrences of 6 and 30 occurrences of 3.

rep(c(4,6,3),times=c(10,20,30))

OUTPUT :



```
seq(1,20)
seq(20,1)
c(1:20,19:1)
seq(5,100,by=5)
rep(c(4,6,3),times=10)
rep(c(4,6,3),times=c(10,20,30))
```

```
> seq(1,20)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> seq(20,1)
 [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
> c(1:20,19:1)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 19 18 17 16 15 14
[27] 13 12 11 10  9  8  7  6  5  4  3  2  1
> seq(5,100,by=5)
 [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95
[20] 100
> rep(c(4,6,3),times=10)
 [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3
> rep(c(4,6,3),times=c(10,20,30))
 [1] 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 3 3 3 3 3 3 3 3 3
[40] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
> 
```

2) Create a vector with elements 1, 2, 3, 4, 5 and call it x.Create another vector with elements 10 20 30 40 50 and call it y What is the value of

a) x + y **b)** x – y **c)** x * y **d)** y /x **e)** x > y **f)** x+2

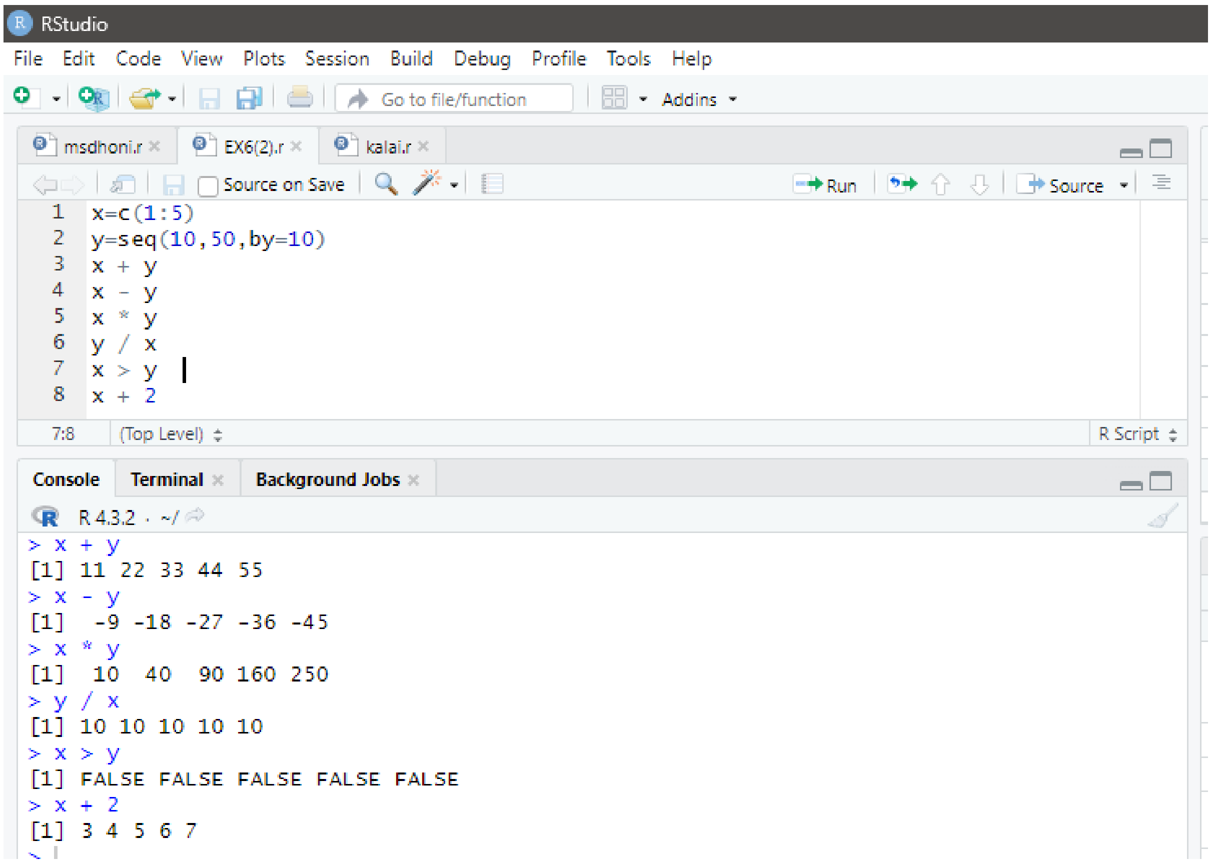
x=c(1:5)

y=seq(10,50,by=10)

x + y   x – y   x * y

y / x   x > y   x + 2

OUTPUT :



```
x=c(1:5)
y=seq(10,50,by=10)
x + y
x - y
x * y
y / x
x > y
x + 2
```

```
> x + y
[1] 11 22 33 44 55
> x - y
[1]  -9 -18 -27 -36 -45
> x * y
[1]  10  40  90 160 250
> y / x
[1] 10 10 10 10 10
> x > y
[1] FALSE FALSE FALSE FALSE FALSE
> x + 2
[1] 3 4 5 6 7
>
```

3) Create a Vector 'v' with the values 10, 20, 30, 10, 40, 50, 30, 40, 80, 90, 100 and do the following exercises.

v = c(10, 20, 30, 10, 40, 50, 30, 40, 80, 90, 100)

a) List elements of the vector that are greater than 10 and less than 80. **v[v>10 & v<80]**

b) List elements of the vector that are multiples of 4.

v[v %% 4 ==0]

c) How many times the element 40 occurred in the vector.

sum (v == 40)

d) List the last value in the given vector. **v[length(v)]**

e) Find second highest value in the given vector.

sort(v , decreasing = TRUE) [2]

f) Find nth highest value in the given vector.

n = 6

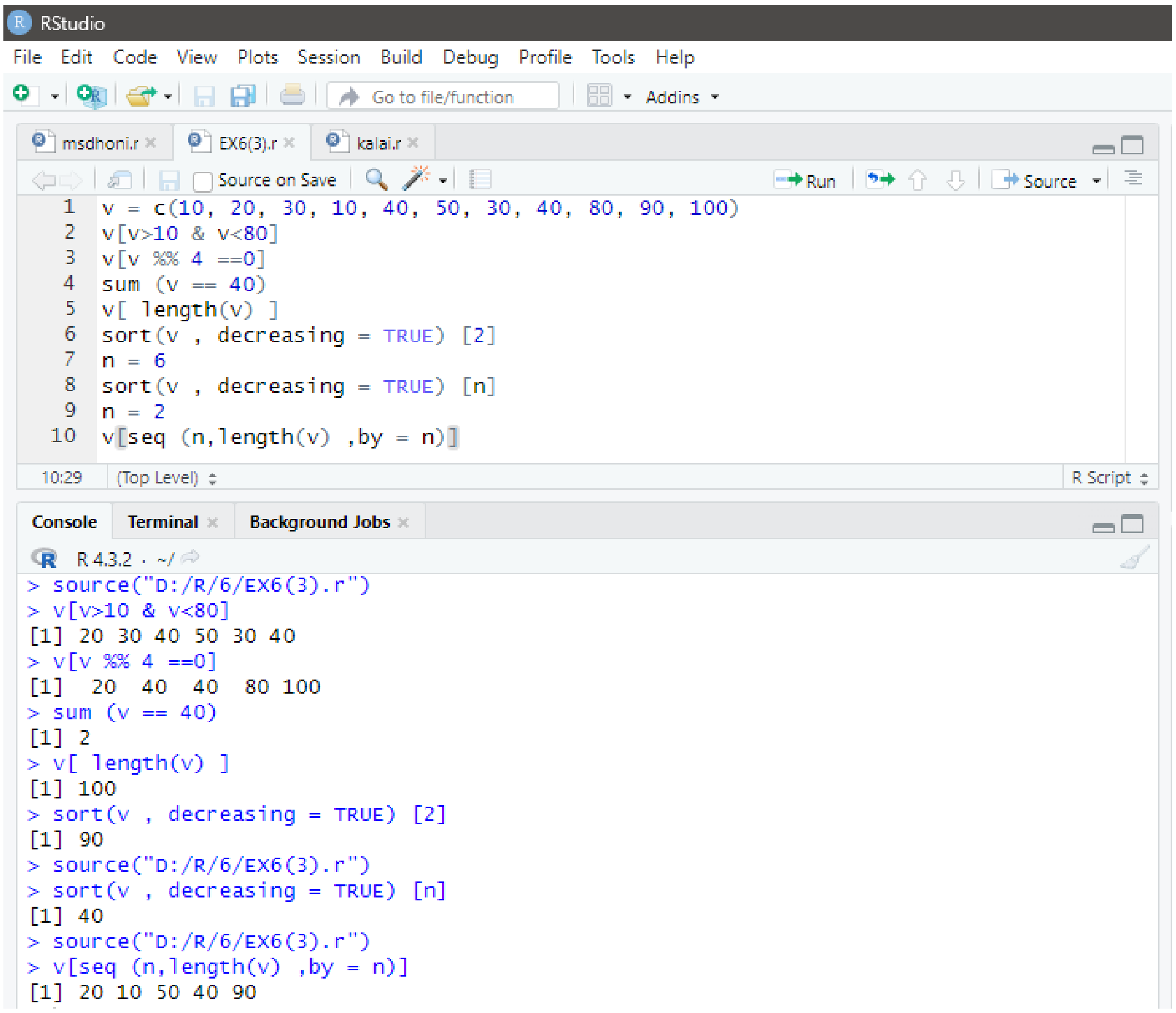
sort(v , decreasing = TRUE) [n]

g) Extract every nth element of a given vector.

n = 2

v[seq (n,length(v) ,by = n)]

OUTPUT :



```
v = c(10, 20, 30, 10, 40, 50, 30, 40, 80, 90, 100)
v[v>10 & v<80]
v[v %% 4 ==0]
sum (v == 40)
v[ length(v) ]
sort(v , decreasing = TRUE) [2]
n = 6
sort(v , decreasing = TRUE) [n]
n = 2
v[seq (n,length(v) ,by = n)]
```

```
> source("D:/R/6/EX6(3).r")
> v[v>10 & v<80]
[1] 20 30 40 50 30 40
> v[v %% 4 ==0]
[1]  20  40  40  80 100
> sum (v == 40)
[1] 2
> v[ length(v) ]
[1] 100
> sort(v , decreasing = TRUE) [2]
[1] 90
> source("D:/R/6/EX6(3).r")
> sort(v , decreasing = TRUE) [n]
[1] 40
> source("D:/R/6/EX6(3).r")
> v[seq (n,length(v) ,by = n)]
[1] 20 10 50 40 90
```

EX.NO : 07

DATE :

MATRIX MANIPULATION

AIM :

To create and manipulate matrix in R.

MATRIX :

Matrix is a two dimensional data structure in R programming. It can be created using the matrix() function. Dimension of the matrix can be defined by passing appropriate value for arguments nrow and ncol

PROGRAM :

1) Create the following matrix.

```
1 4
2 5
3 6
```

```
matrix1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3) print(matrix1)
```

2) Create a vector v of size 12 with random value between 1 to 100.

```
v = sample(1:100,12) print(v)
```

3) Convert the vector v to a 4*3 matrix1.

```
vec <- c(16, 5, 4, 3, 2, 1)

matrix1 <- matrix(vec,nrow = 2, ncol = 3) print(matrix1)
```

4) Change the column names of matrix1 to x, y, z and row names to a, b, c, d.

```
rownames(matrix1) = c("a","b")

colnames(matrix1) = c("x","y","z") print(matrix1)
```

5) Compute A+3, A-3, A*3 and A/3.

```
matrix1+3 matrix1-3 matrix1*3 matrix1/3
```

6) Obtain the transpose matrix of matrix1.

```
t (matrix1)
```

7) Display 2nd row of matrix.

```
Matrix1 [2,  ]
```

8) Display the entire matrix leaving 2nd column.

```
Matrix1 [ , -2]
```

9) Display only first three rows of matrix.

```
Matrix1 [ c(1:3), ]
```

10) Change the value of element at 2nd row and 3rd column to 300.

```
Matrix1 [2:3] = 300
```

11) Replace all elements of matrix1 that are greater than 50 with 200.

```
Matrix1 [ matrix1>50]=200
```

12) Display column and row names of matrix1.

```
rownames (matrix1 ) colnames(matrix1)
```

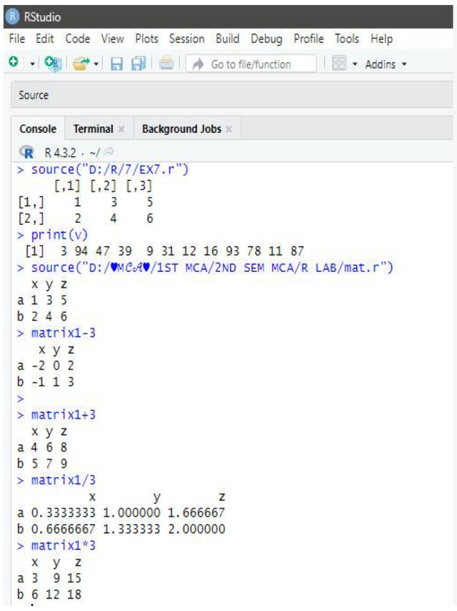
13) Display dimension and the no of elements in matrix1.

```
dim(matrix1) length(matrix1)
```

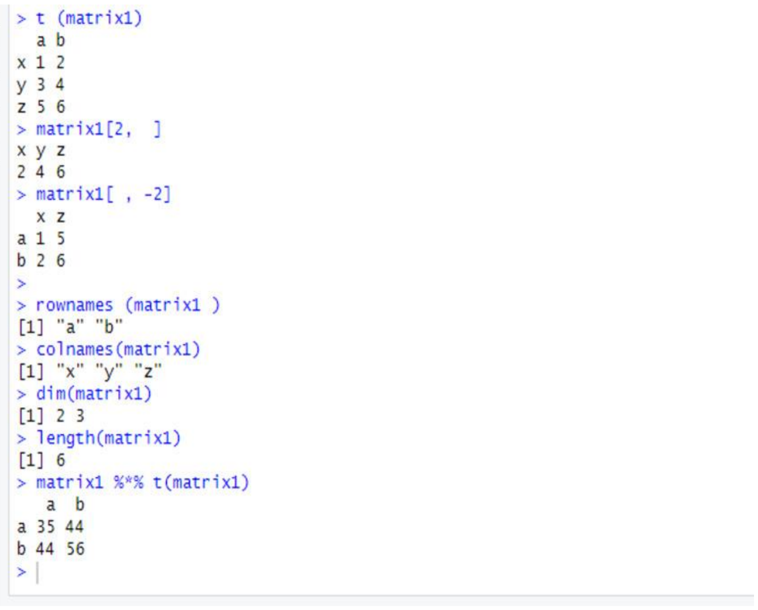
14) Multiply the matrix1 with its transpose.

```
Matrix1 %*% t(matrix1)
```

OUTPUT :



```
> source("D:/R/7/EX7.r")
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> print(v)
 [1]  3 94 47 39  9 31 12 16 93 78 11 87
> source("D:/♥MCA♥/1ST MCA/2ND SEM MCA/R LAB/mat.r")
  x y z
a 1 3 5
b 2 4 6
> matrix1-3
   x y z
a -2 0 2
b -1 1 3
>
> matrix1+3
  x y z
a 4 6 8
b 5 7 9
> matrix1/3
          x        y        z
a 0.3333333 1.000000 1.666667
b 0.6666667 1.333333 2.000000
> matrix1*3
  x  y  z
a 3  9 15
b 6 12 18
```



```
> t (matrix1)
  a b
x 1 2
y 3 4
z 5 6
> matrix1[2,  ]
x y z
2 4 6
> matrix1[ , -2]
  x z
a 1 5
b 2 6
>
> rownames (matrix1 )
[1] "a" "b"
> colnames(matrix1)
[1] "x" "y" "z"
> dim(matrix1)
[1] 2 3
> length(matrix1)
[1] 6
> matrix1 %*% t(matrix1)
   a  b
a 35 44
b 44 56
> 
```

EX.NO : 08

DATE :

LIST MANIPULATION

AIM :

To create and manipulate list in R.

LIST :

List is a data structure having components of mixed data types. A vector having all elements of the same type is called atomic vector but a vector havingelements of different type is called list. It can be created using the list() function.

PROGRAM :

1) Create a list called my_list from the 3 vectors provided below and do the following.

```
a <- "My First List"

b <- c("R", "is", "Fun!")

c <- matrix(1:9,3,3)

Mylist = list(a,b,c) print(Mylist)
```

a) Access second component of my_list.

```
Mylist[2]
```

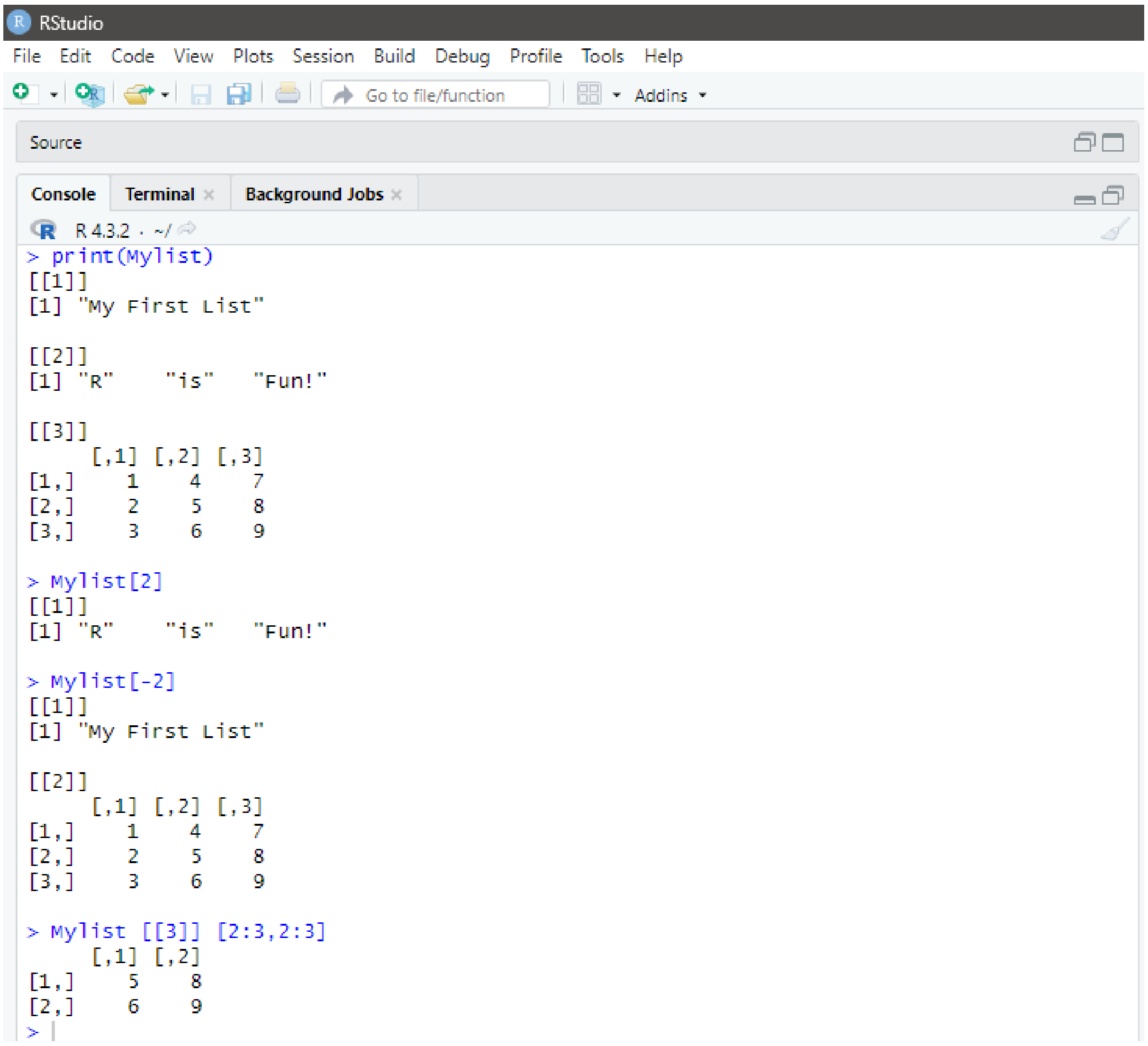
b) Display all components of my_list leaving second component.

```
Mylist[-2]
```

c) Remove the first and third items from my_list.

```
Mylist [[3]] [2:3,2:3]
```

OUTPUT :


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Background Jobs
R 4.3.2 · ~/

```
> print(Mylist)
[[1]]
[1] "My First List"

[[2]]
[1] "R"     "is"    "Fun!"

[[3]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> Mylist[2]
[[1]]
[1] "R"     "is"    "Fun!"

> Mylist[-2]
[[1]]
[1] "My First List"

[[2]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> Mylist [[3]] [2:3,2:3]
     [,1] [,2]
[1,]    5    8
[2,]    6    9
> 
```

2) Make a list called Stuff and display its structure. The list should contain three elements.

   a) The sequence of even numbers from 4 to 100. Its name should be even.

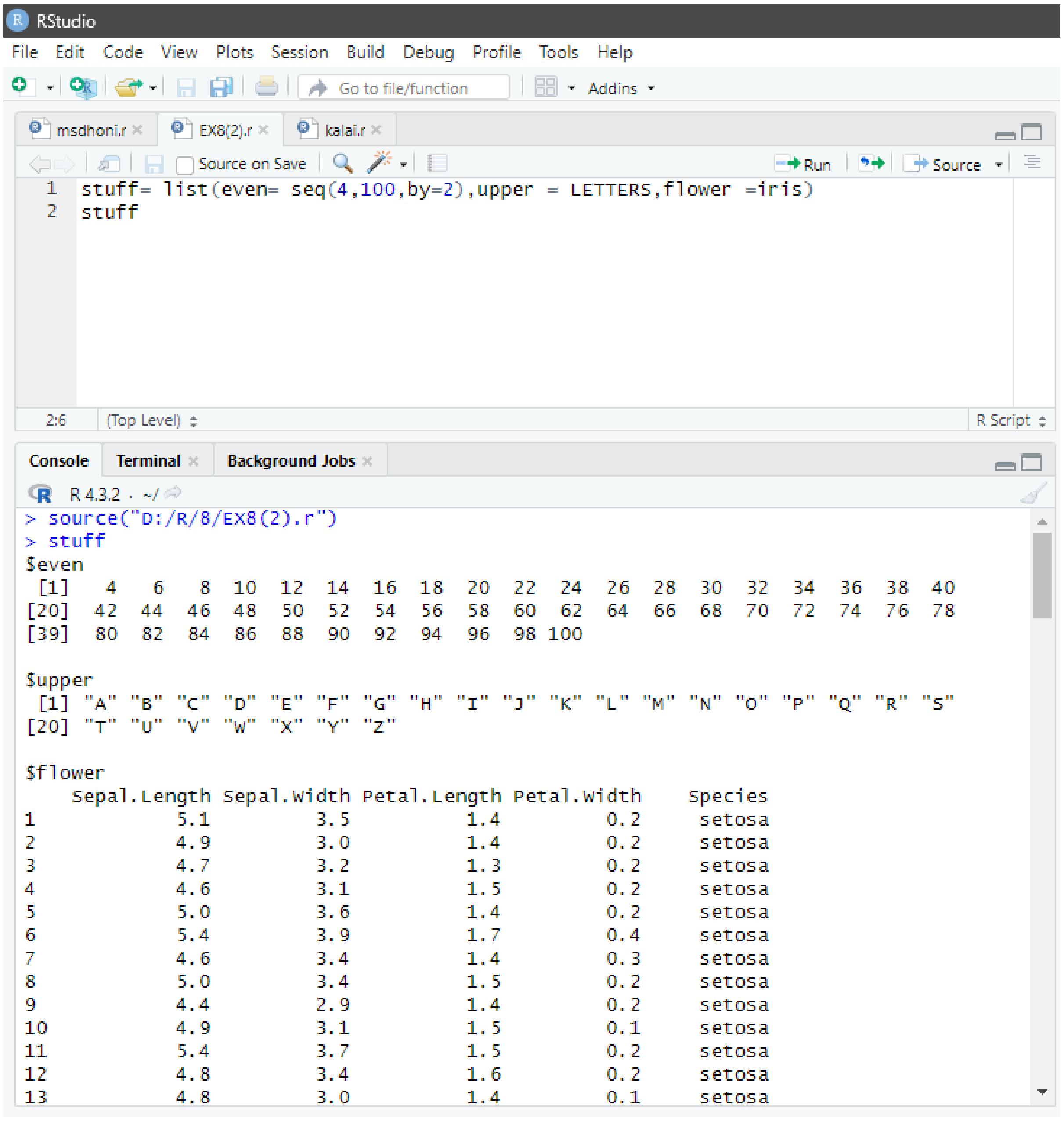
   b) The uppercase letters of the alphabet. Its name should be upper.

   c) The data frame iris. Its name should be flower.

stuff= list(even= seq(4,100,by=2),upper = LETTERS,flower =iris)

stuff

OUTPUT :



```
stuff= list(even= seq(4,100,by=2),upper = LETTERS,flower =iris)
stuff
```

```
> source("D:/R/8/EX8(2).r")
> stuff
$even
 [1]   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40
[20]  42  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78
[39]  80  82  84  86  88  90  92  94  96  98 100

$upper
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"

$flower
    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
```

EX.NO : 09

DATE :

DATA FRAMES IN R

AIM :

To create data frame and perform various operations on data frame in R.

DATA FRAMES :

Data frame is a two dimensional data structure in R. It is a special case of a list which has each component of equal length. Each component forms the column and contents of the component form the rows.

PROGRAM :

1) Create data frame to display voting details.

```
dataframe1 <- data.frame ( Name = c("Kalai", "Divya", "Pavai"), Age = c(22, 15, 9), Vote = c(TRUE, FALSE, TRUE))

print(dataframe1)
```

2) Access the name column from data frame.

```
# pass index number inside [ ].

    print(dataframe1[1])

# pass column name inside [[ ]].

    print(dataframe1[["Name"]])

# use $ operator and column name.

    print(dataframe1$Name)
```

3) Use rbind() in data frame.

```
dataframe2 <- data.frame (Name = c("Basheer", "Jothi"), Age = c(46, 89), Vote = c(TRUE,TRUE))
```

```
dataframe1 <- data.frame ( Name = c("Kalai", "Divya", "Pavai"), Age = c(22, 15, 9), Vote = c(TRUE, FALSE, TRUE))

updated <- rbind( dataframe1, dataframe2)

print(updated)
```

4) Use cbind() in data frame.

# create a data frame.

```
dataframe1 <- data.frame ( Name = c("Kalai", "Divya"), Age = c(22, 15))
```

# create another data frame.

```
dataframe2 <- data.frame ( Hobby = c("Cricket", "Anime"))
```

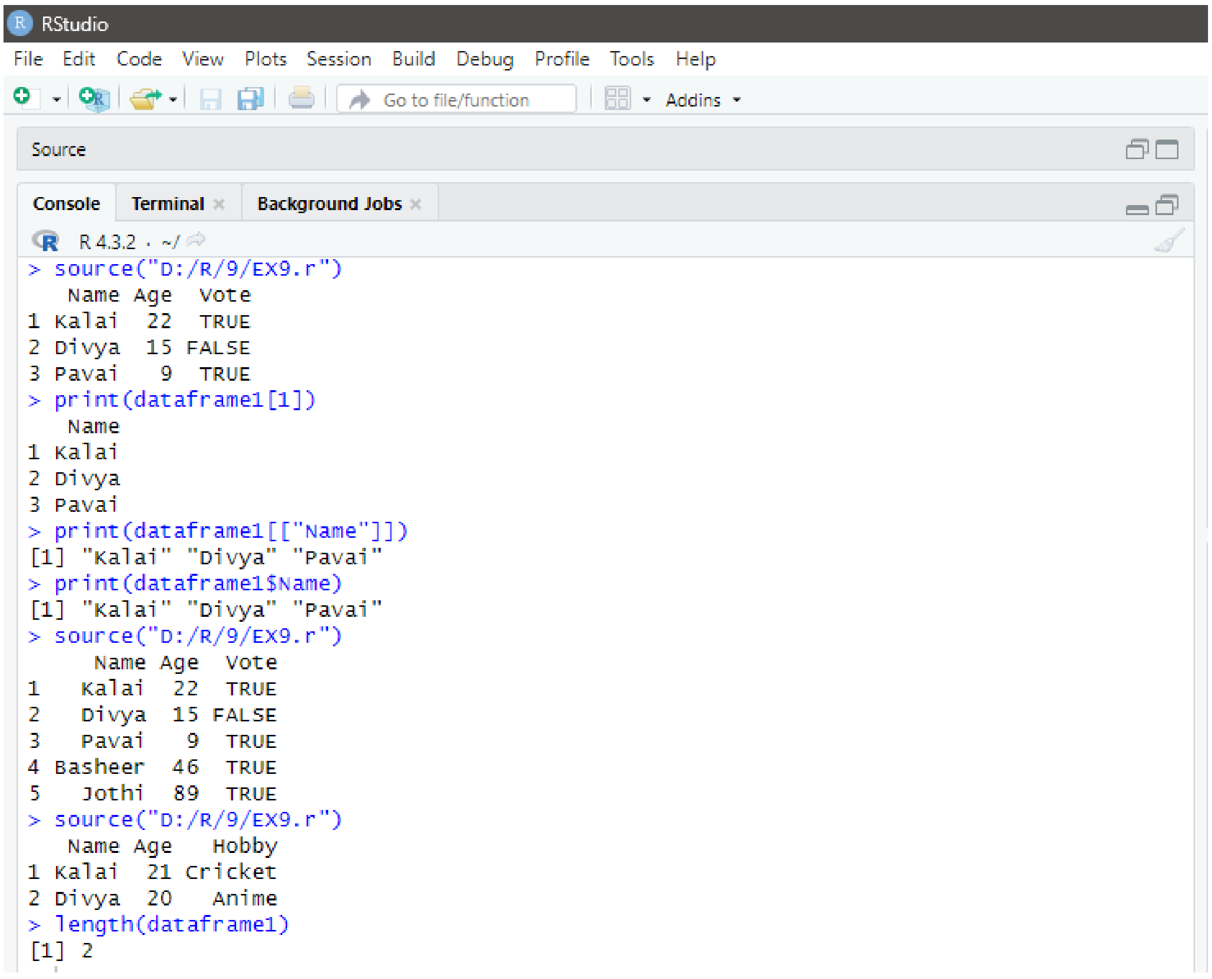
# combine two data frames horizontally.

```
updated <- cbind(dataframe1, dataframe2)

print(updated)
```

5) Find the length of Data frame.

```
length(dataframe1)
```

OUTPUT :



```
> source("D:/R/9/EX9.r")
   Name Age  Vote
1 Kalai  22  TRUE
2 Divya  15 FALSE
3 Pavai   9  TRUE
> print(dataframe1[1])
   Name
1 Kalai
2 Divya
3 Pavai
> print(dataframe1[["Name"]])
[1] "Kalai" "Divya" "Pavai"
> print(dataframe1$Name)
[1] "Kalai" "Divya" "Pavai"
> source("D:/R/9/EX9.r")
     Name Age  Vote
1   Kalai  22  TRUE
2   Divya  15 FALSE
3   Pavai   9  TRUE
4 Basheer  46  TRUE
5   Jothi  89  TRUE
> source("D:/R/9/EX9.r")
   Name Age   Hobby
1 Kalai  21 Cricket
2 Divya  20   Anime
> length(dataframe1)
[1] 2
```

EX.NO : 10

DATE :

IMPORTING FILES IN R

AIM :

To import different types of file such as csv,xml, excel and text files in R.

PROGRAM :

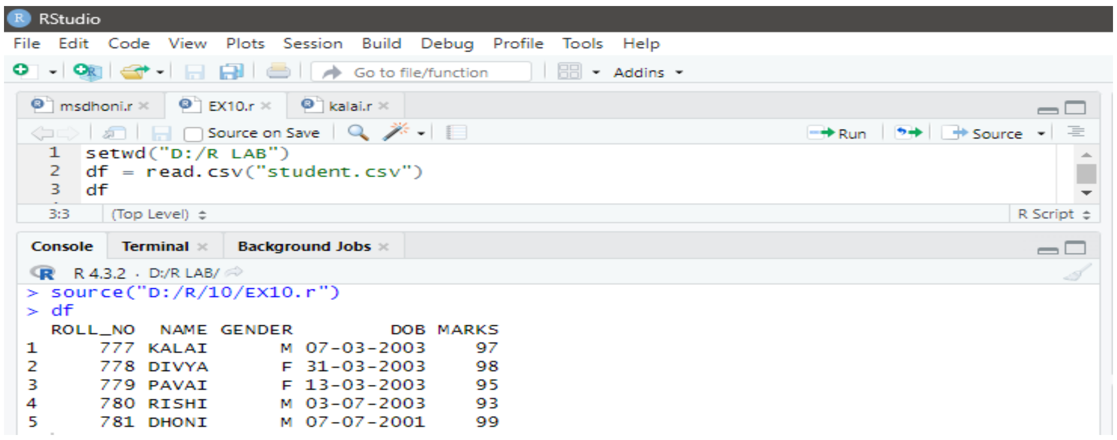
1) Importing a CSV File :-

Create a csv file with the following student data and save it as student.csv, Import this file in r studio.

ROLL_NO	NAME	GENDER	DOB	MARKS
777	KALAI	M	07-03-2003	97
778	DIVYA	F	31-03-2003	98
779	PAVAI	F	13-03-2003	95
780	RISHI	M	03-07-2003	93
781	DHONI	M	07-07-2001	99

setwd(“D:/R LAB”)
df = read.csv(“student.csv”)   df

OUTPUT :



```
setwd("D:/R LAB")
df = read.csv("student.csv")
df
```

```
> source("D:/R/10/EX10.r")
> df
  ROLL_NO  NAME GENDER        DOB MARKS
1     777 KALAI      M 07-03-2003    97
2     778 DIVYA      F 31-03-2003    98
3     779 PAVAI      F 13-03-2003    95
4     780 RISHI      M 03-07-2003    93
5     781 DHONI      M 07-07-2001    99
```

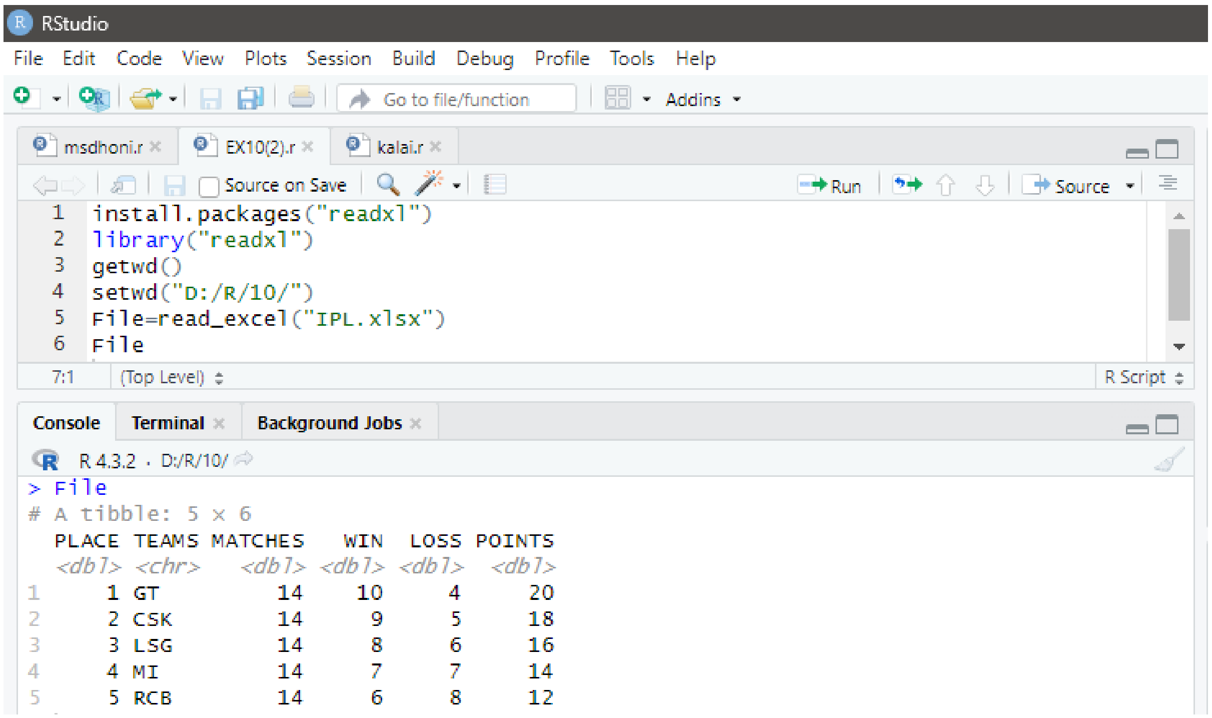
2) Importing a Excel File.

Create a excel file which contains the following data and save it as ipl.xlsx.

PLACE	TEAMS	MATCHES	WIN	LOSS	POINTS
1	GT	14	10	4	20
2	CSK	14	9	5	18
3	LSG	14	8	6	16
4	MI	14	7	7	14
5	RCB	14	6	8	12

```
install.packages("readxl")
library("readxl")
getwd()
setwd("D:/R/10/")
File=read_excel("IPL.xlsx") File
```

OUTPUT :



```
install.packages("readxl")
library("readxl")
getwd()
setwd("D:/R/10/")
File=read_excel("IPL.xlsx")
File
```

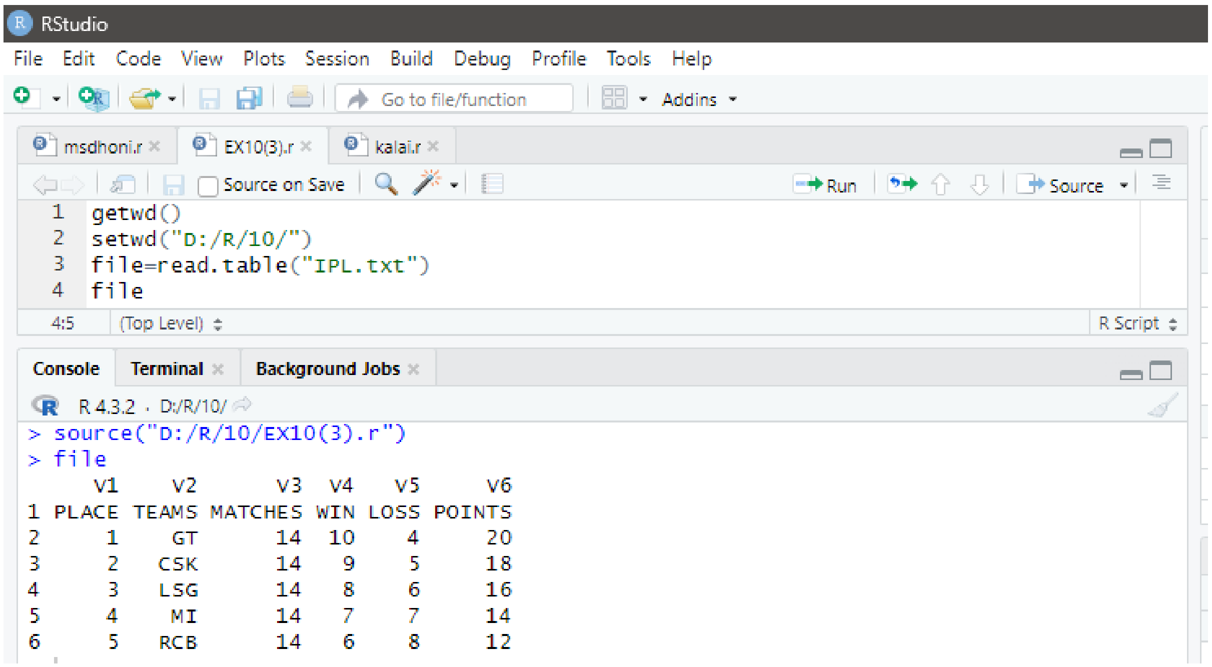
```
> File
# A tibble: 5 × 6
  PLACE TEAMS MATCHES   WIN  LOSS POINTS
  <dbl> <chr>   <dbl> <dbl> <dbl>  <dbl>
1     1 GT         14    10     4     20
2     2 CSK        14     9     5     18
3     3 LSG        14     8     6     16
4     4 MI         14     7     7     14
5     5 RCB        14     6     8     12
```

3) Importing a Text File.

Create a text file using notepad with the following data and save it as ‘IPL.txt’.

PLACE	TEAMS	MATCHES	WIN	LOSS	POINTS
1	GT	14	10	4	20
2	CSK	14	9	5	18
3	LSG	14	8	6	16
4	MI	14	7	7	14
5	RCB	14	6	8	12

OUTPUT :



```
getwd()
setwd("D:/R/10/")
file=read.table("IPL.txt")
file
```

```
> source("D:/R/10/EX10(3).r")
> file
     V1    V2      V3  V4   V5     V6
1 PLACE TEAMS MATCHES WIN LOSS POINTS
2     1    GT      14  10    4     20
3     2   CSK      14   9    5     18
4     3   LSG      14   8    6     16
5     4    MI      14   7    7     14
6     5   RCB      14   6    8     12
```

EX.NO : 11

DATE :

IMPLEMENTING NAVIE BAYES USING IRIS DATASET

AIM :

To show the implementation of Navie Bayes algorithm using Iris dataset.

NAVIE BAYES ALGORITHM :

Naive Bayes is a Supervised Machine Learning algorithm based on the Bayes Theorem that is used to solve classification problems by following a probabilistic approach. It is based on the idea that the predictor variables in a Machine Learning model are independent of each other. Meaning that the outcome of a model depends on a set of independent variables(Predictor) that have nothing to do with each other.

PROCEDURE :

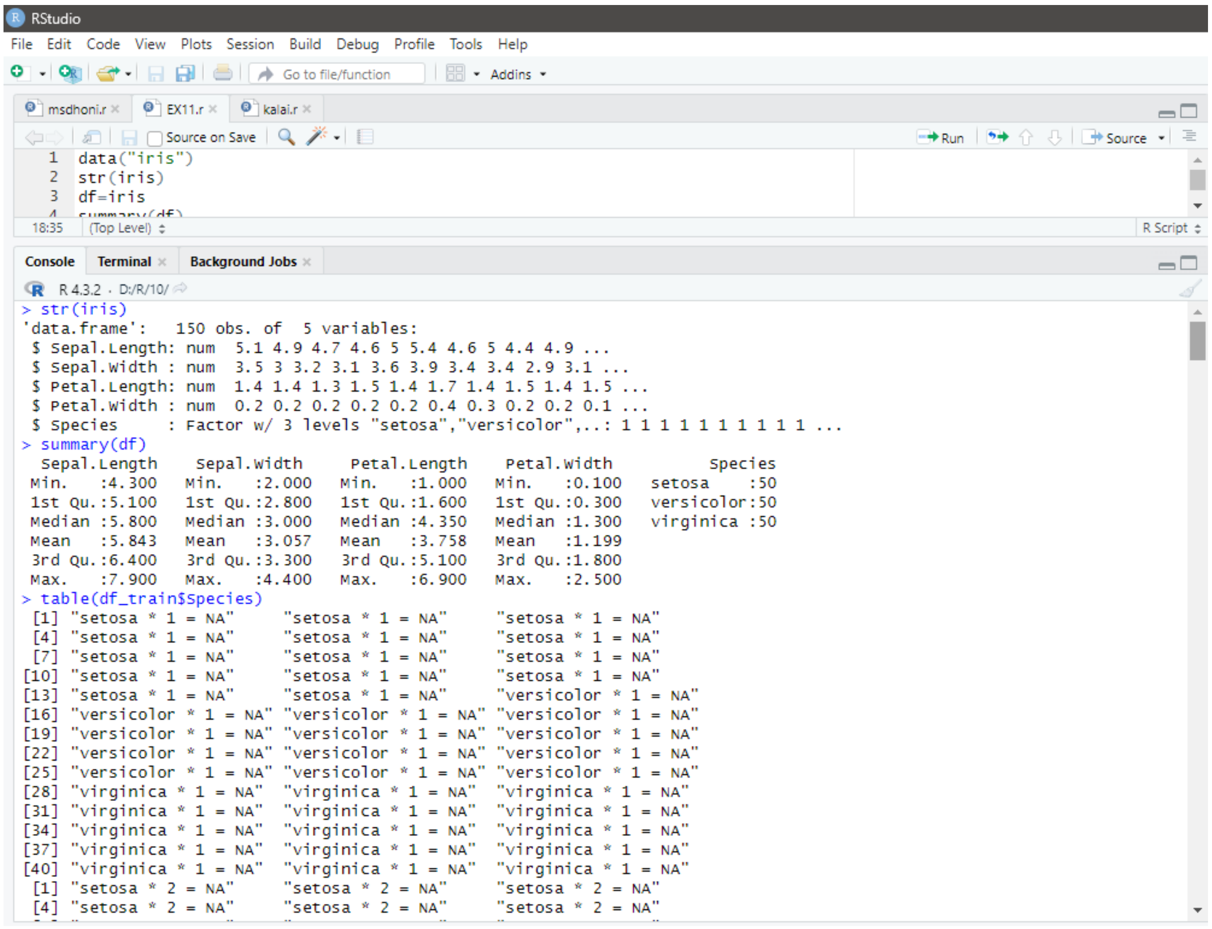
1) Load the dataset .
2) Create train/test set.
  a) id <- sample(2,nrow(df),replace=TRUE,prob=c(0.80,0.20))
  b) df_train <- df [id==1,]
  c) df_test <- df [id==2,]
3) Build the model.
  a) Install Package e1071
  b) Load the package
  c) Model <- naviesBayes(df_train[,5] ,df_train$species)
4) Make prediction.
  a) p= predict(model,df_test[, -5])
5) Measure performance by confusion matrix.

PROGRAM :

```
data("iris")
str(iris)
df=iris
```

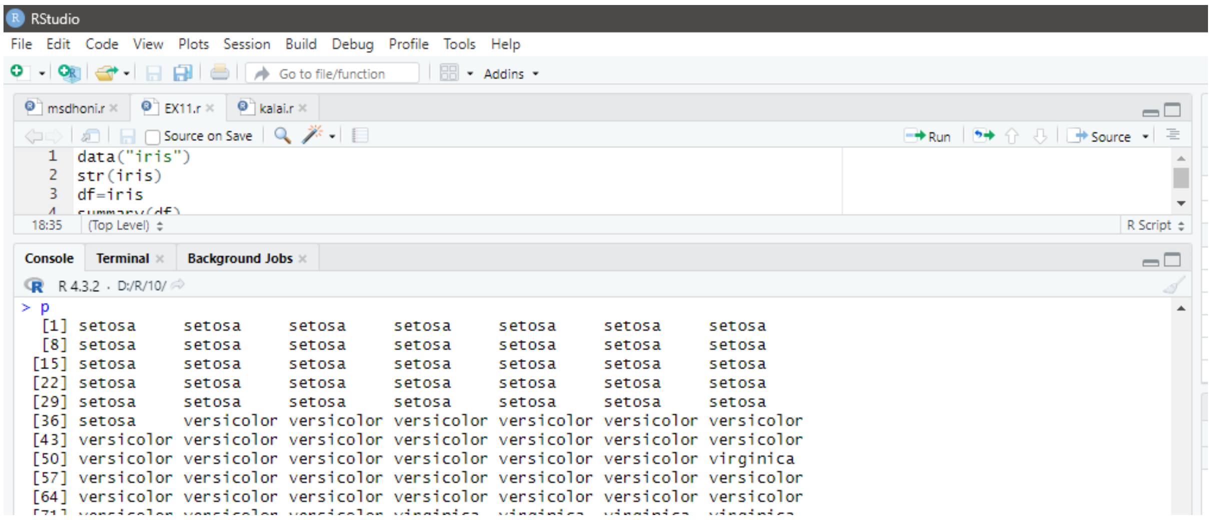
```
summary(df)
set.seed(123)
id = sample(1:2,nrow(df),replace=TRUE,prob=c(0.08,0.20))
df_train = df[id == 1,]
df_test = df[id == 2,]
table(df_train$Species)
table(df_test$Species)
install.packages("e1071")
library(e1071)
model=naiveBayes(df_train[,-5],df_train$Species)
p<- predict(model,df_test[,-5])
p
install.packages("caret")
library(caret)
confusionMatrix(df_test$Species,p)
```

OUTPUT :



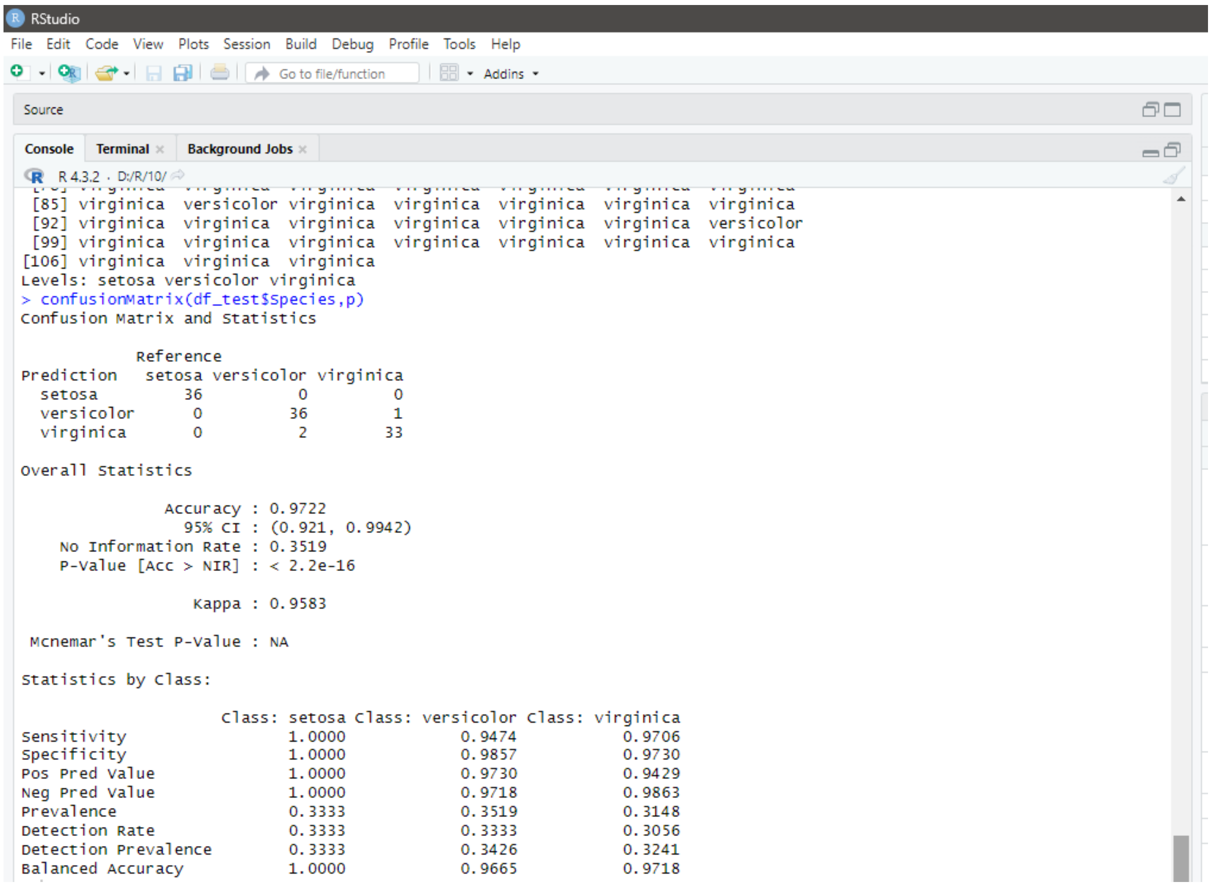
```
data("iris")
str(iris)
df=iris
summary(df)
```

```
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> summary(df)
  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                  
> table(df_train$Species)
 [1] "setosa * 1 = NA"     "setosa * 1 = NA"     "setosa * 1 = NA"    
 [4] "setosa * 1 = NA"     "setosa * 1 = NA"     "setosa * 1 = NA"    
 [7] "setosa * 1 = NA"     "setosa * 1 = NA"     "setosa * 1 = NA"    
[10] "setosa * 1 = NA"     "setosa * 1 = NA"     "setosa * 1 = NA"    
[13] "setosa * 1 = NA"     "setosa * 1 = NA"     "versicolor * 1 = NA"
[16] "versicolor * 1 = NA" "versicolor * 1 = NA" "versicolor * 1 = NA"
[19] "versicolor * 1 = NA" "versicolor * 1 = NA" "versicolor * 1 = NA"
[22] "versicolor * 1 = NA" "versicolor * 1 = NA" "versicolor * 1 = NA"
[25] "versicolor * 1 = NA" "versicolor * 1 = NA" "versicolor * 1 = NA"
[28] "virginica * 1 = NA"  "virginica * 1 = NA"  "virginica * 1 = NA" 
[31] "virginica * 1 = NA"  "virginica * 1 = NA"  "virginica * 1 = NA" 
[34] "virginica * 1 = NA"  "virginica * 1 = NA"  "virginica * 1 = NA" 
[37] "virginica * 1 = NA"  "virginica * 1 = NA"  "virginica * 1 = NA" 
[40] "virginica * 1 = NA"  "virginica * 1 = NA"  "virginica * 1 = NA" 
 [1] "setosa * 2 = NA"     "setosa * 2 = NA"     "setosa * 2 = NA"    
 [4] "setosa * 2 = NA"     "setosa * 2 = NA"     "setosa * 2 = NA"    
```



```
data("iris")
str(iris)
df=iris
summary(df)
```

```
> p
  [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
  [8] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
 [15] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
 [22] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
 [29] setosa     setosa     setosa     setosa     setosa     setosa     setosa    
 [36] setosa     versicolor versicolor versicolor versicolor versicolor versicolor
 [43] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [50] versicolor versicolor versicolor versicolor versicolor versicolor virginica 
 [57] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [64] versicolor versicolor versicolor versicolor versicolor versicolor versicolor
 [71] versicolor versicolor versicolor virginica  virginica  virginica  virginica 
```



```
 [85] virginica  versicolor virginica  virginica  virginica  virginica  virginica 
 [92] virginica  virginica  virginica  virginica  virginica  virginica  versicolor
 [99] virginica  virginica  virginica  virginica  virginica  virginica  virginica 
[106] virginica  virginica  virginica 
Levels: setosa versicolor virginica
> confusionMatrix(df_test$Species,p)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         36          0         0
  versicolor      0         36         1
  virginica       0          2        33

Overall Statistics
                                          
               Accuracy : 0.9722          
                 95% CI : (0.921, 0.9942)
    No Information Rate : 0.3519          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.9583          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9474           0.9706
Specificity                 1.0000            0.9857           0.9730
Pos Pred Value              1.0000            0.9730           0.9429
Neg Pred Value              1.0000            0.9718           0.9863
Prevalence                  0.3333            0.3519           0.3148
Detection Rate              0.3333            0.3333           0.3056
Detection Prevalence        0.3333            0.3426           0.3241
Balanced Accuracy           1.0000            0.9665           0.9718
```

EX.NO : 12

DATE :

BREAST CANCER PREDICTION USING KNN

AIM :

To show Implementation of KNN using Breast Cancer dataset.

KNN ALGORITHM :

K Nearest Neighbors or KNN Algorithm is a simple algorithm which uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction.

PROCEDURE :

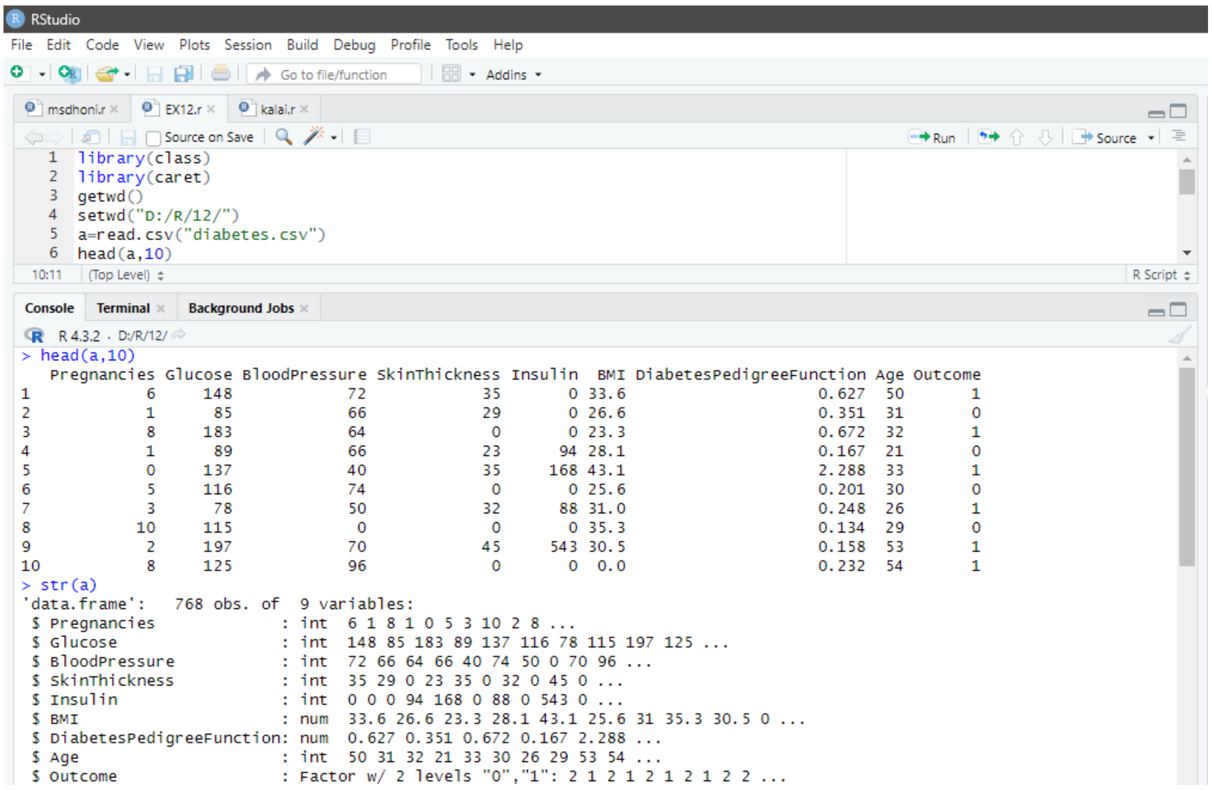
```
1) Load the dataset.
2) Initialize the value of k.
3) For each sample in the training dataset.
    a) Calculate the distance between test data and each sample in
       the training data.
4) Based on the distance value, sort them in ascending order.
5) Get top k rows from the sorted array.
6) Return the most frequent class of these rows.
```

PROGRAM :

```
library(class)

library(caret)

a=read.csv("diabetes.csv")

head(a,10)

str(a)

a$Outcome=factor(a$Outcome)
```

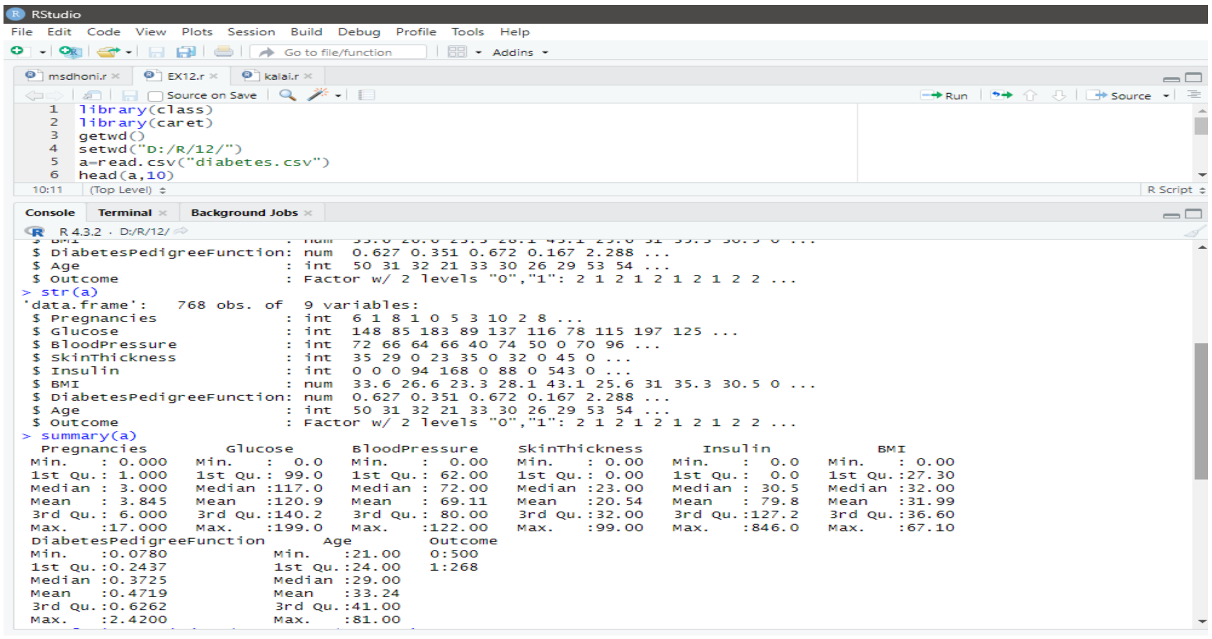
```
str(a)
summary(a)        #a[r,c]
is.na(a)
colSums(is.na(a))
train=a[1:500,]
test=a[501:768,]
# knn(training,test)
pred_test=knn(train[,-9],test[,-9],train$Outcome,k=7)
cm=table(pred_test,test$Outcome)
cm
confusionMatrix(pred_test,test$Outcome)
```

OUTPUT :



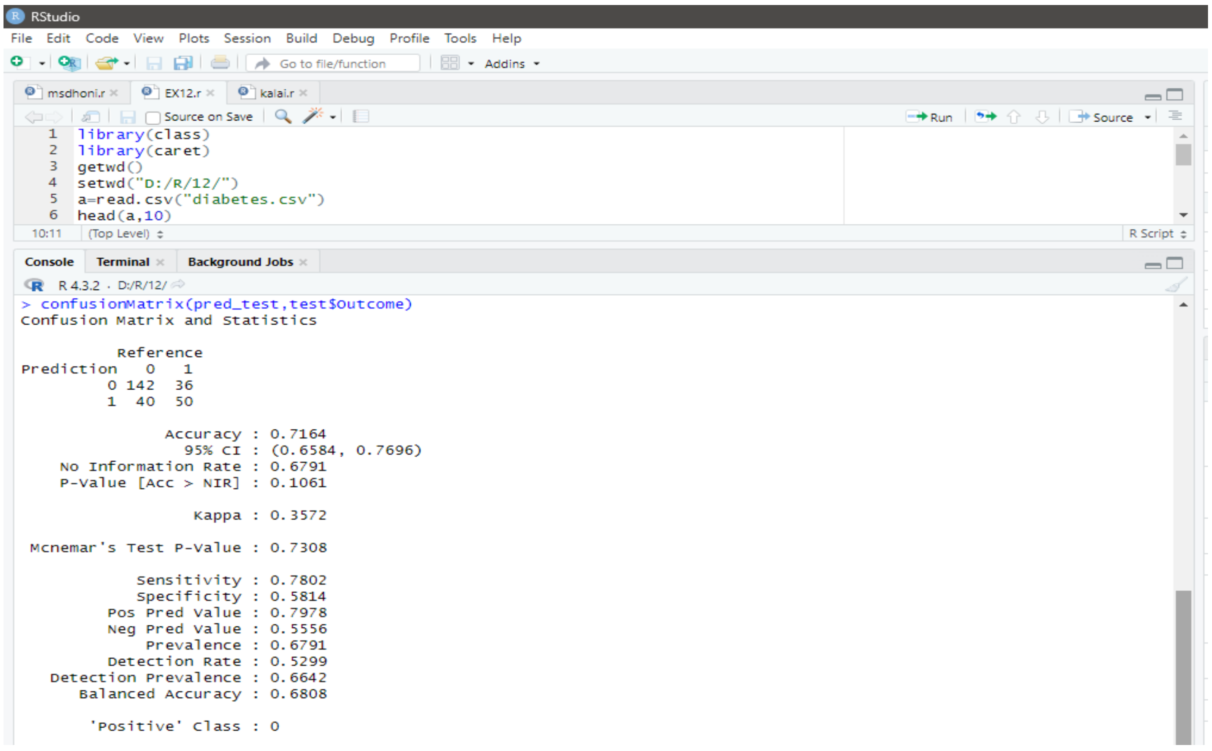
```
library(class)
library(caret)
getwd()
setwd("D:/R/12/")
a=read.csv("diabetes.csv")
head(a,10)
```

```
> head(a,10)
   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction Age Outcome
1            6     148            72            35       0 33.6                    0.627  50       1
2            1      85            66            29       0 26.6                    0.351  31       0
3            8     183            64             0       0 23.3                    0.672  32       1
4            1      89            66            23      94 28.1                    0.167  21       0
5            0     137            40            35     168 43.1                    2.288  33       1
6            5     116            74             0       0 25.6                    0.201  30       0
7            3      78            50            32      88 31.0                    0.248  26       1
8           10     115             0             0       0 35.3                    0.134  29       0
9            2     197            70            45     543 30.5                    0.158  53       1
10           8     125            96             0       0  0.0                    0.232  54       1
> str(a)
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2 ...
```



```r
library(class)
library(caret)
getwd()
setwd("D:/R/12/")
a=read.csv("diabetes.csv")
head(a,10)
```

```
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2 ...
> str(a)
'data.frame':	768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2 ...
> summary(a)
  Pregnancies        Glucose       BloodPressure    SkinThickness      Insulin           BMI       
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00  
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30  
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00  
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99  
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60  
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10  
 DiabetesPedigreeFunction      Age        Outcome
 Min.   :0.0780           Min.   :21.00   0:500  
 1st Qu.:0.2437           1st Qu.:24.00   1:268  
 Median :0.3725           Median :29.00          
 Mean   :0.4719           Mean   :33.24          
 3rd Qu.:0.6262           3rd Qu.:41.00          
 Max.   :2.4200           Max.   :81.00          
```



```r
library(class)
library(caret)
getwd()
setwd("D:/R/12/")
a=read.csv("diabetes.csv")
head(a,10)
```

```
> confusionMatrix(pred_test,test$Outcome)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 142  36
         1  40  50
                                          
               Accuracy : 0.7164          
                 95% CI : (0.6584, 0.7696)
    No Information Rate : 0.6791          
    P-Value [Acc > NIR] : 0.1061          
                                          
                  Kappa : 0.3572          
                                          
 Mcnemar's Test P-Value : 0.7308          
                                          
            Sensitivity : 0.7802          
            Specificity : 0.5814          
         Pos Pred Value : 0.7978          
         Neg Pred Value : 0.5556          
             Prevalence : 0.6791          
         Detection Rate : 0.5299          
   Detection Prevalence : 0.6642          
      Balanced Accuracy : 0.6808          
                                          
       'Positive' Class : 0               
```

EX.NO : 13

DATE :

DIABETIES PREDICTION USING DECISION TREE

AIM :

To show the implementation of Decision Tree using Diabeties dataset.

DECISION TREE ALGORITHM :

Decision Tree is one of the most widely used and practical methods for supervised learning.They can be used to solve both regression and classification problems. It is robust to noisy data and capable of learning disjunctive expressions. The decision tree algorithms such as ID3, CART, C4.5 & ASSISTANT are very popular inductive inference algorithms, and they are successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

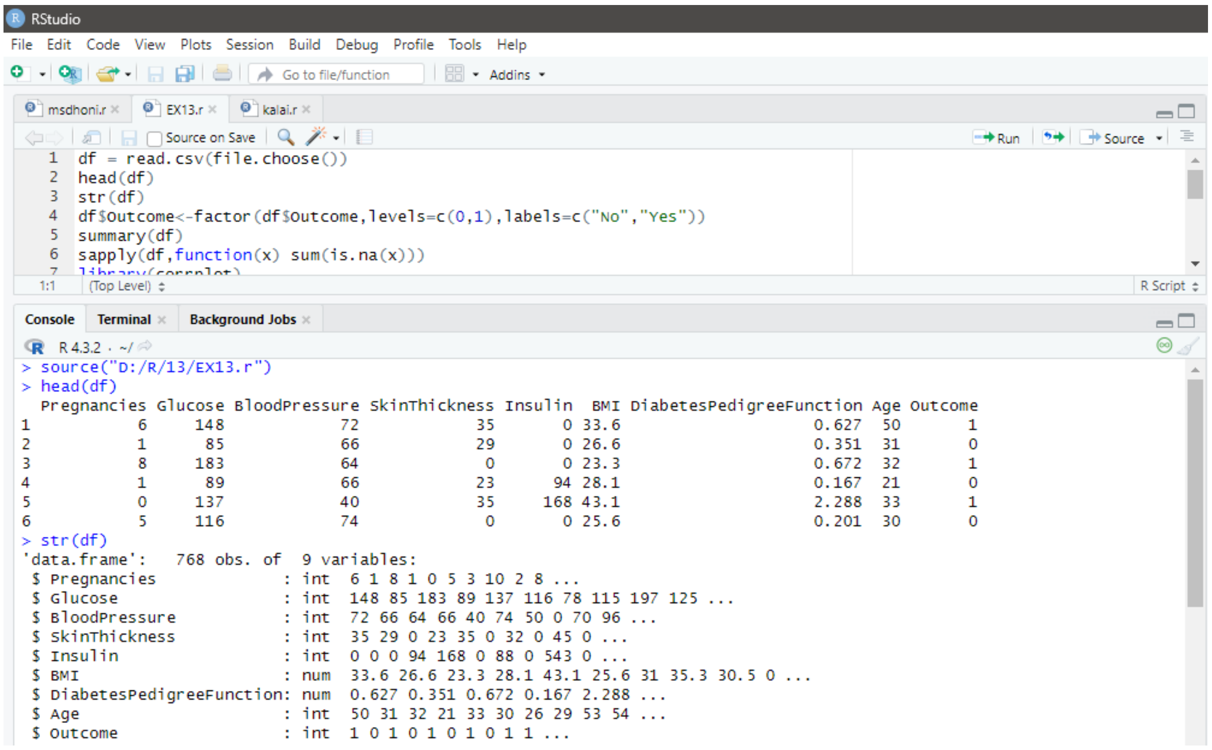
PROCEDURE :

1) Load the dataset
2) Create train/test set.
  a) train <- sample(n,trunc(0.70*n))
  b) df_train <- df [train,]
  c) df_test <- df [- train,]
3) Build the model.
  a) Install Package rpart
  b) Load the package .
  c) Model <- rpart(Outcome~. ,data =df_train)
4) Make prediction.
  a) predict (model ,df_test,type=”class”)
5) Measure performance by confusion matrix.

PROGRAM :

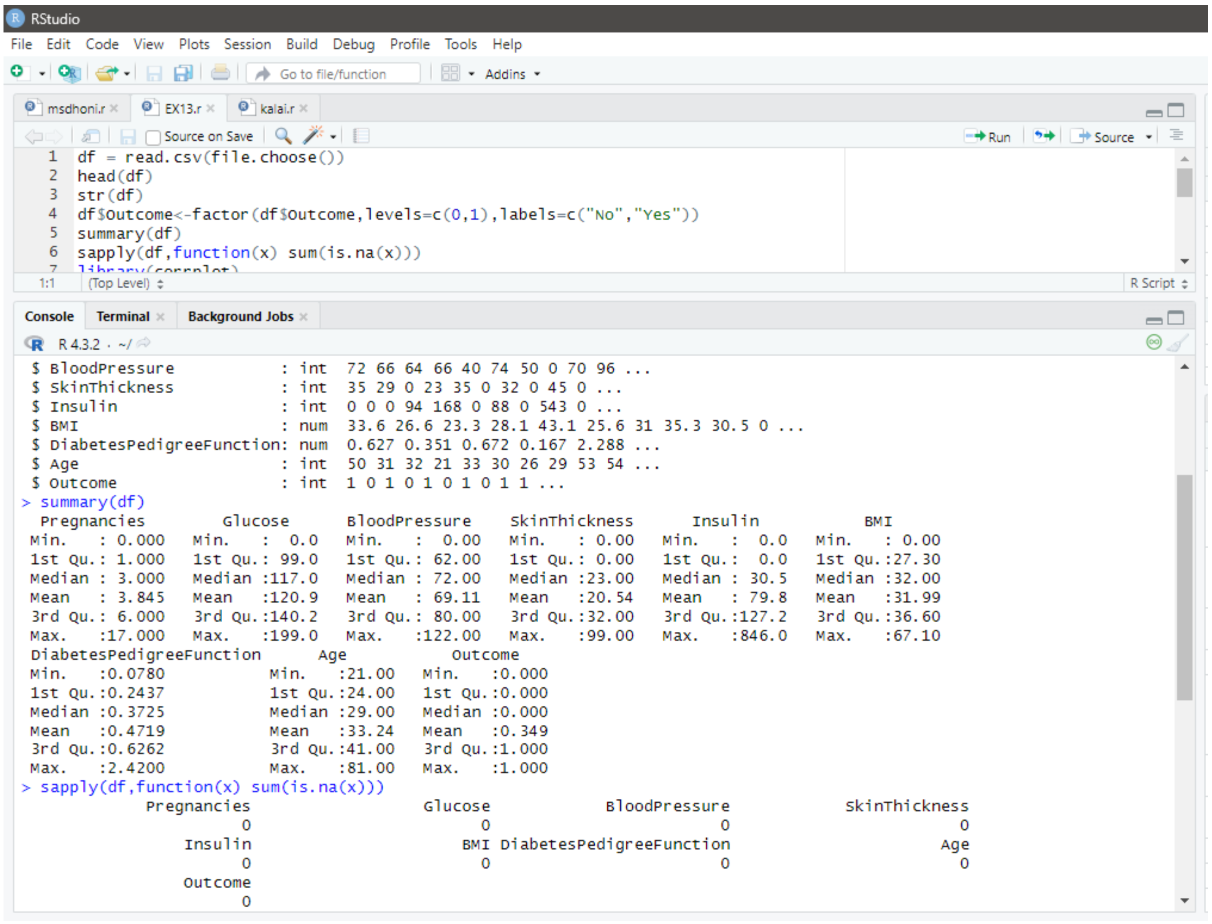
```
df = read.csv(file.choose())

head(df)

str(df)

df$Outcome<-
factor(df$Outcome,levels=c(0,1),labels=c("No","Yes"))

summary(df)

sapply(df,function(x) sum(is.na(x))) library(corrplot)

set.seed(123)

n<-nrow(df)

train<-sample(n,trunc(0.70*n))

df_train<-df[train,]

df_test<-df[-train,]

library(rpart)

model<-rpart(Outcome~.,data=df_train)

plot(model,margin=0.01)

text(model,use.n=TRUE,pretty=TRUE,cex=0.8)

p<-predict(model,df_test,type="class")

library(caret)

confusionMatrix(p,df_test$Outcome)
```

OUTPUT :



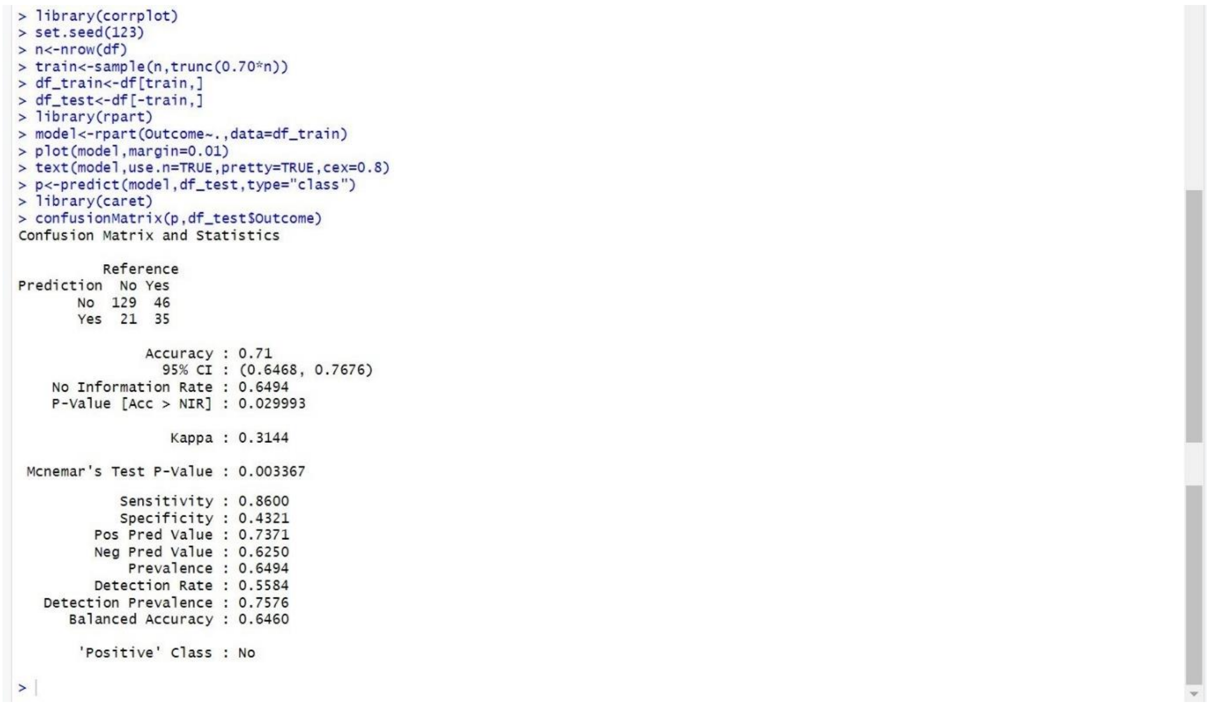
```
df = read.csv(file.choose())
head(df)
str(df)
df$Outcome<-factor(df$Outcome,levels=c(0,1),labels=c("No","Yes"))
summary(df)
sapply(df,function(x) sum(is.na(x)))
library(corrplot)
```

```
> source("D:/R/13/EX13.r")
> head(df)
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI DiabetesPedigreeFunction Age Outcome
1           6     148            72            35       0 33.6                    0.627  50       1
2           1      85            66            29       0 26.6                    0.351  31       0
3           8     183            64             0       0 23.3                    0.672  32       1
4           1      89            66            23      94 28.1                    0.167  21       0
5           0     137            40            35     168 43.1                    2.288  33       1
6           5     116            74             0       0 25.6                    0.201  30       0
> str(df)
'data.frame':	768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

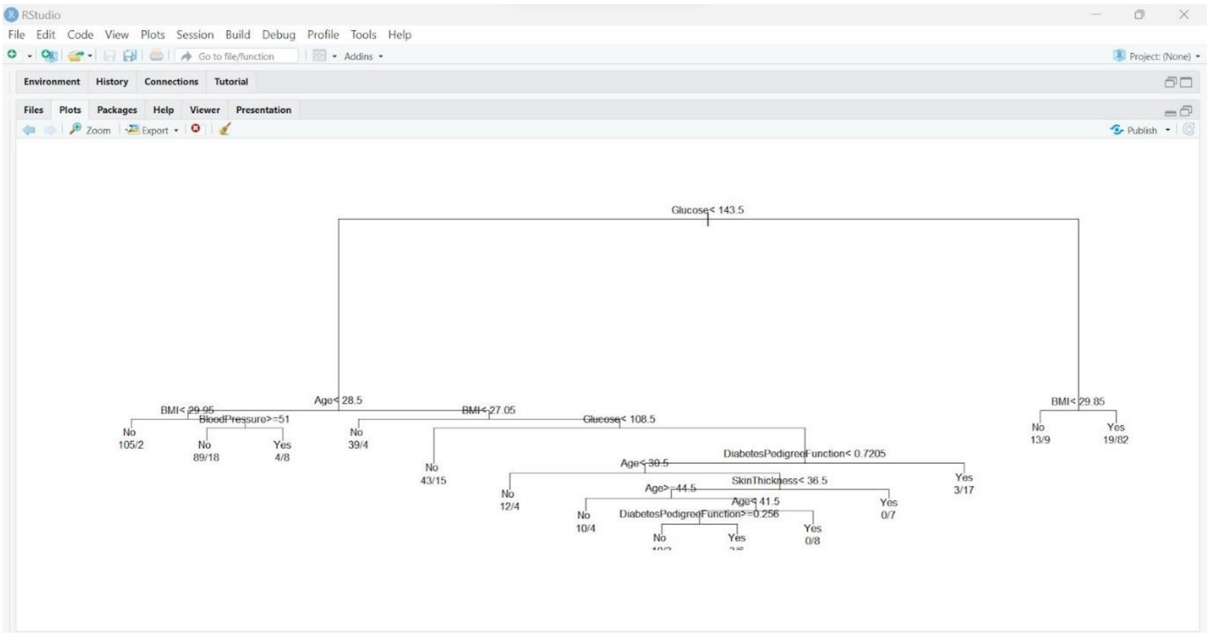


```
df = read.csv(file.choose())
head(df)
str(df)
df$Outcome<-factor(df$Outcome,levels=c(0,1),labels=c("No","Yes"))
summary(df)
sapply(df,function(x) sum(is.na(x)))
library(corrplot)
```

```
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
> summary(df)
  Pregnancies        Glucose      BloodPressure    SkinThickness      Insulin           BMI       
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00  
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30  
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00  
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99  
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60  
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10  
 DiabetesPedigreeFunction      Age           Outcome     
 Min.   :0.0780           Min.   :21.00   Min.   :0.000  
 1st Qu.:0.2437           1st Qu.:24.00   1st Qu.:0.000  
 Median :0.3725           Median :29.00   Median :0.000  
 Mean   :0.4719           Mean   :33.24   Mean   :0.349  
 3rd Qu.:0.6262           3rd Qu.:41.00   3rd Qu.:1.000  
 Max.   :2.4200           Max.   :81.00   Max.   :1.000  
> sapply(df,function(x) sum(is.na(x)))
             Pregnancies                  Glucose            BloodPressure            SkinThickness 
                       0                        0                        0                        0 
                 Insulin                      BMI DiabetesPedigreeFunction                      Age 
                       0                        0                        0                        0 
                 Outcome 
                       0 
```



```
> library(corrplot)
> set.seed(123)
> n<-nrow(df)
> train<-sample(n,trunc(0.70*n))
> df_train<-df[train,]
> df_test<-df[-train,]
> library(rpart)
> model<-rpart(Outcome~.,data=df_train)
> plot(model,margin=0.01)
> text(model,use.n=TRUE,pretty=TRUE,cex=0.8)
> p<-predict(model,df_test,type="class")
> library(caret)
> confusionMatrix(p,df_test$Outcome)
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  129  46
       Yes  21  35

               Accuracy : 0.71
                 95% CI : (0.6468, 0.7676)
    No Information Rate : 0.6494
    P-Value [Acc > NIR] : 0.029993

                  Kappa : 0.3144

 Mcnemar's Test P-Value : 0.003367

            Sensitivity : 0.8600
            Specificity : 0.4321
         Pos Pred Value : 0.7371
         Neg Pred Value : 0.6250
             Prevalence : 0.6494
         Detection Rate : 0.5584
   Detection Prevalence : 0.7576
      Balanced Accuracy : 0.6460

       'Positive' Class : No

> 
```



EX.NO : 14

DATE :

WINE QUALITY PREDICTION USING RANDOM FOREST

AIM :

To show the Implementation of Random Forest using Wine Quality Prediction dataset.

RANDOM FOREST ALGORITHM :

Random forest is a popular supervised machine learning algorithm—used for both classification and regression problems. It is based on the concept of ensemble learning, which enables users to combine multiple classifiers to solve a complex problem and to also improve the performance of the model. The random forest algorithm relies on multiple decision trees and accepts the results of the predictions from each tree. Based on the majority votes of predictions, it determines the final result.

PROCEDURE :

1) Load the dataset.
2) Create train/test set.
  a) id <- sample(2,nrow(df),replace=TRUE,prob=c(0.70,0.30))
  b) df_train <- df [id==1,]
  c) df_test <- df [id==2,]
3) Build the model.
  a) Install Package randomForest.
  b) Load the package .
  c) Model<-randomForest(taste~.,data=df_train,ntree=1000,ntry=5)
4) Make prediction.

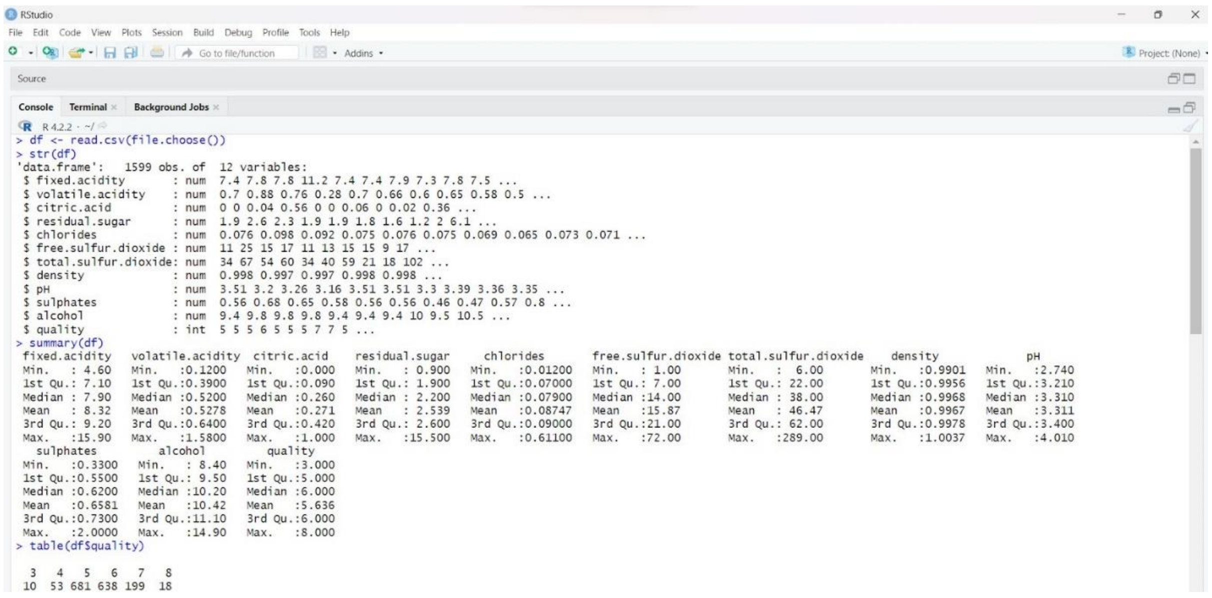
  p= predict(model,df_test)

5) Calculate the votes for each of the predicted targets.
6) The most highly voted predicted target is the final prediction.

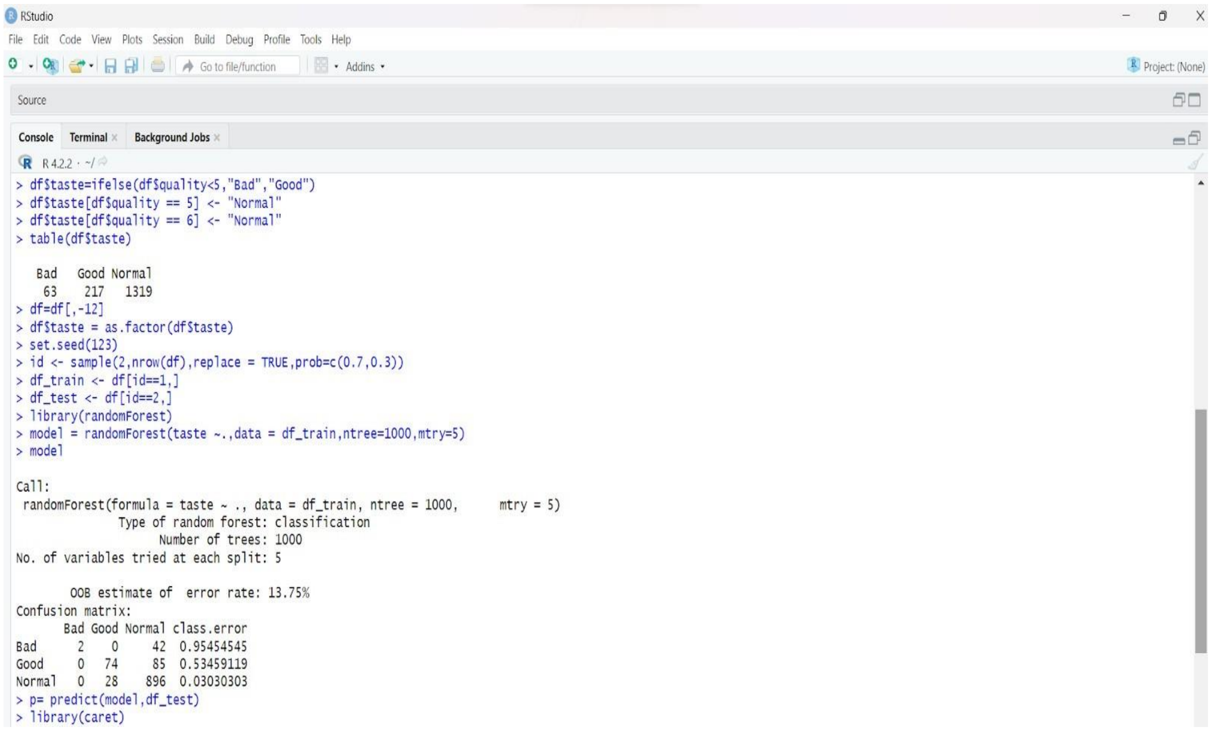
PROGRAM :

```
df <- read.csv(file.choose())

str(df)

summary(df)

table(df$quality)

df$taste=ifelse(df$quality<5,"Bad","Good")

df$taste[df$quality == 5] <- "Normal"

df$taste[df$quality == 6] <- "Normal"

table(df$taste)

df=df[,-12]

df$taste = as.factor(df$taste)

set.seed(123)

id <- sample(2,nrow(df),replace = TRUE,prob=c(0.7,0.3))

df_train <- df[id==1,]

df_test <- df[id==2,]

install.packages("randomForest")

library(randomForest)

model = randomForest(taste ~.,data = df_train,ntree=1000,mtry=5)

model

p= predict(model,df_test)

library(caret)

confusionMatrix(p,df_test$taste)
```

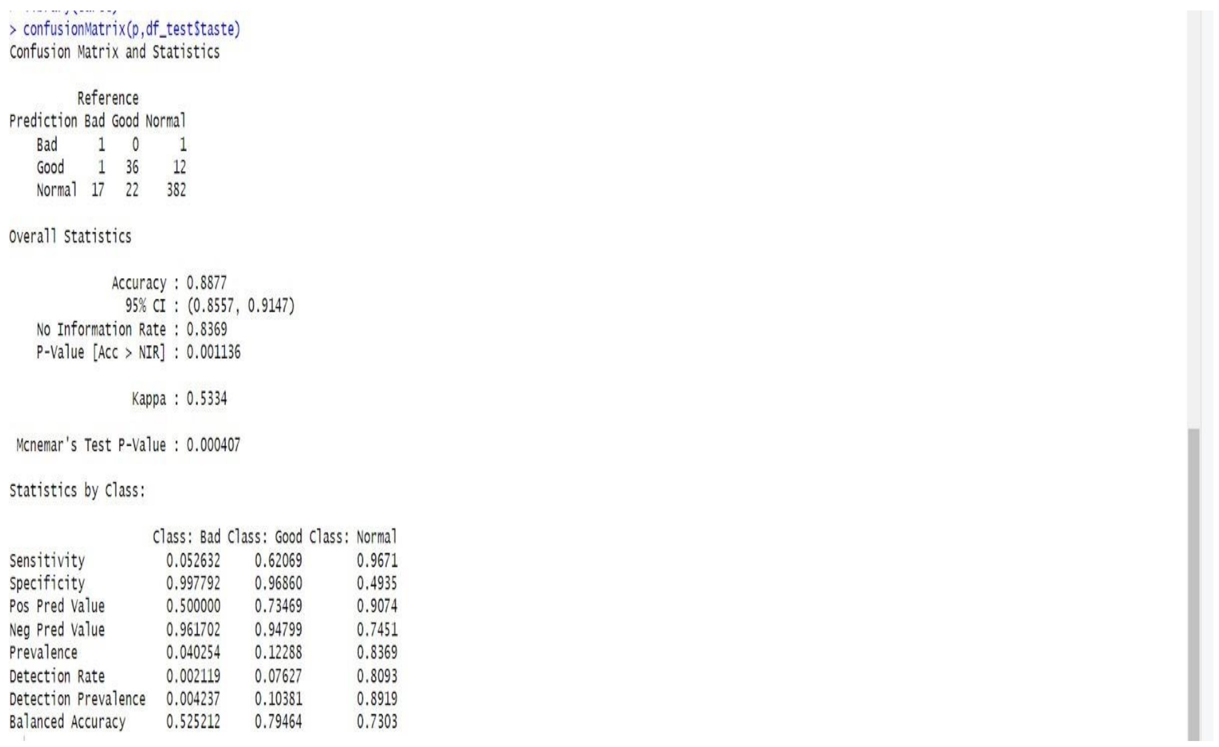
OUTPUT :



```
> df <- read.csv(file.choose())
> str(df)
'data.frame':	1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
> summary(df)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides       free.sulfur.dioxide total.sulfur.dioxide    density             pH       
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901   Min.   :2.740  
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210  
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968   Median :3.310  
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967   Mean   :3.311  
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978   3rd Qu.:3.400  
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037   Max.   :4.010  
   sulphates         alcohol          quality     
 Min.   :0.3300   Min.   : 8.40   Min.   :3.000  
 1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000  
 Median :0.6200   Median :10.20   Median :6.000  
 Mean   :0.6581   Mean   :10.42   Mean   :5.636  
 3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000  
 Max.   :2.0000   Max.   :14.90   Max.   :8.000  
> table(df$quality)

  3   4   5   6   7   8 
 10  53 681 638 199  18 
```



```
> df$taste=ifelse(df$quality<5,"Bad","Good")
> df$taste[df$quality == 5] <- "Normal"
> df$taste[df$quality == 6] <- "Normal"
> table(df$taste)

   Bad   Good Normal 
    63    217   1319 
> df=df[,-12]
> df$taste = as.factor(df$taste)
> set.seed(123)
> id <- sample(2,nrow(df),replace = TRUE,prob=c(0.7,0.3))
> df_train <- df[id==1,]
> df_test <- df[id==2,]
> library(randomForest)
> model = randomForest(taste ~.,data = df_train,ntree=1000,mtry=5)
> model

Call:
 randomForest(formula = taste ~ ., data = df_train, ntree = 1000,      mtry = 5) 
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 5

        OOB estimate of  error rate: 13.75%
Confusion matrix:
       Bad Good Normal class.error
Bad      2    0     42  0.95454545
Good     0   74     85  0.53459119
Normal   0   28    896  0.03030303
> p= predict(model,df_test)
> library(caret)
```



```
> confusionMatrix(p,df_test$taste)
Confusion Matrix and Statistics

          Reference
Prediction Bad Good Normal
    Bad      1    0      1
    Good     1   36     12
    Normal  17   22    382

Overall Statistics

               Accuracy : 0.8877
                 95% CI : (0.8557, 0.9147)
    No Information Rate : 0.8369
    P-Value [Acc > NIR] : 0.001136

                  Kappa : 0.5334

 Mcnemar's Test P-Value : 0.000407

Statistics by Class:

                     Class: Bad Class: Good Class: Normal
Sensitivity            0.052632     0.62069        0.9671
Specificity            0.997792     0.96860        0.4935
Pos Pred Value         0.500000     0.73469        0.9074
Neg Pred Value         0.961702     0.94799        0.7451
Prevalence             0.040254     0.12288        0.8369
Detection Rate         0.002119     0.07627        0.8093
Detection Prevalence   0.004237     0.10381        0.8919
Balanced Accuracy      0.525212     0.79464        0.7303
```

EX.NO : 15

DATE :

CLUSTERING USARREST DATASET USING KMEANS ALGORITHM

AIM :

To show the implementation of KMeans Algorithm using USArrest Dataset.

KMEANS ALGORITHM :

Kmeans clustering is one of the simplest and popular unsupervised machine learning algorithms. A cluster refers to a collection of data points aggregated together because of certain similarities. The number k refers to the number of centroids. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to the nearest cluster.

PROCEDURE :

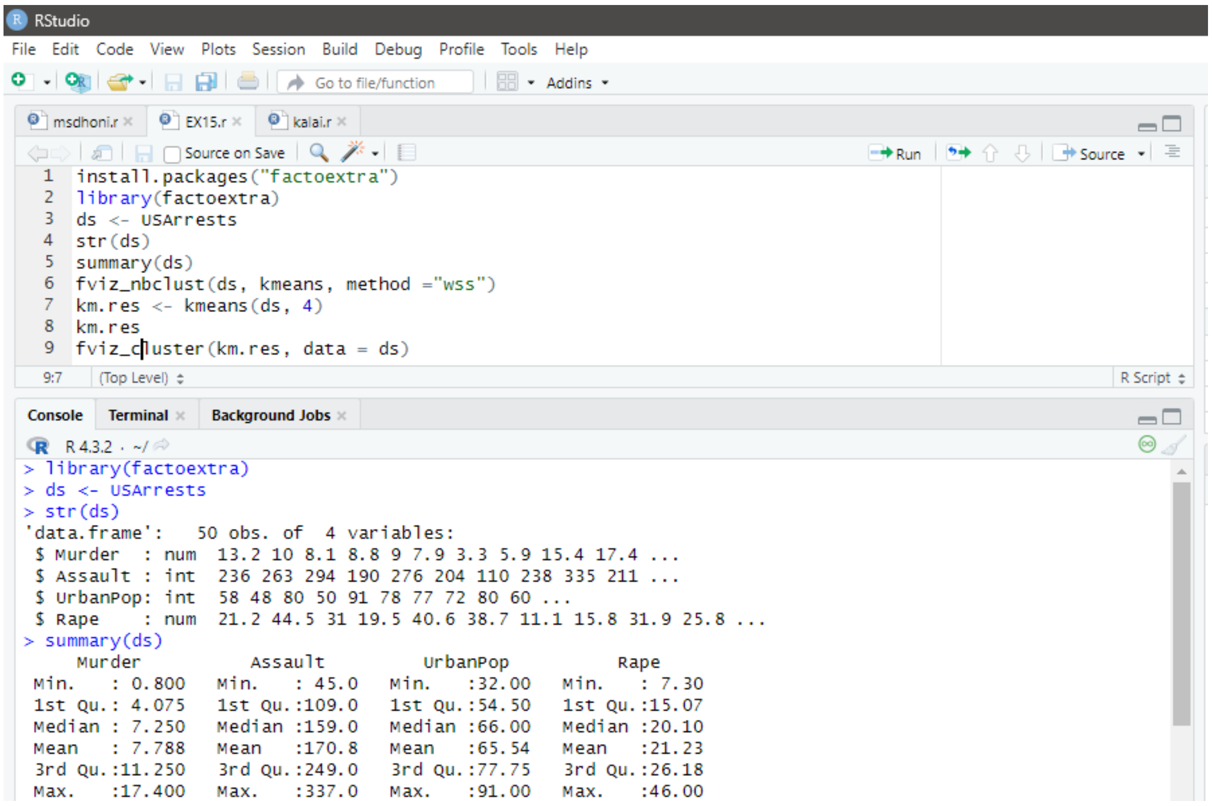
1) Load the dataset.
2) Choose the number K clusters.
3) Select at random K points.
4) Assign each data point to closest centroid that forms K clusters.
5) Compute and place the new centroid of each centroid.
6) Reassign each data point to new cluster.

PROGRAM :

```
install.packages("factoextra")

library(factoextra)

ds <- USArrests

str(ds)

summary(ds)

fviz_nbclust(ds, kmeans, method ="wss")
```

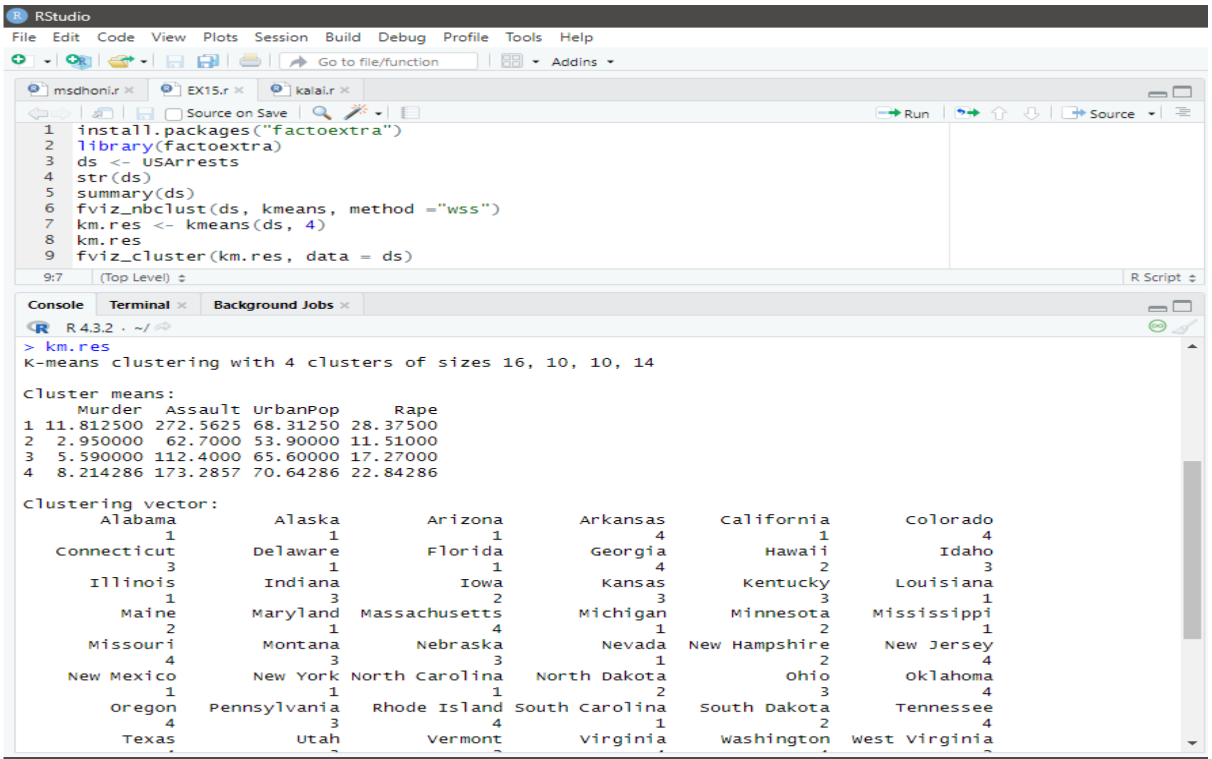
```
km.res <- kmeans(ds, 4)

km.res

fviz_cluster(km.res, data = ds)
```

OUTPUT :



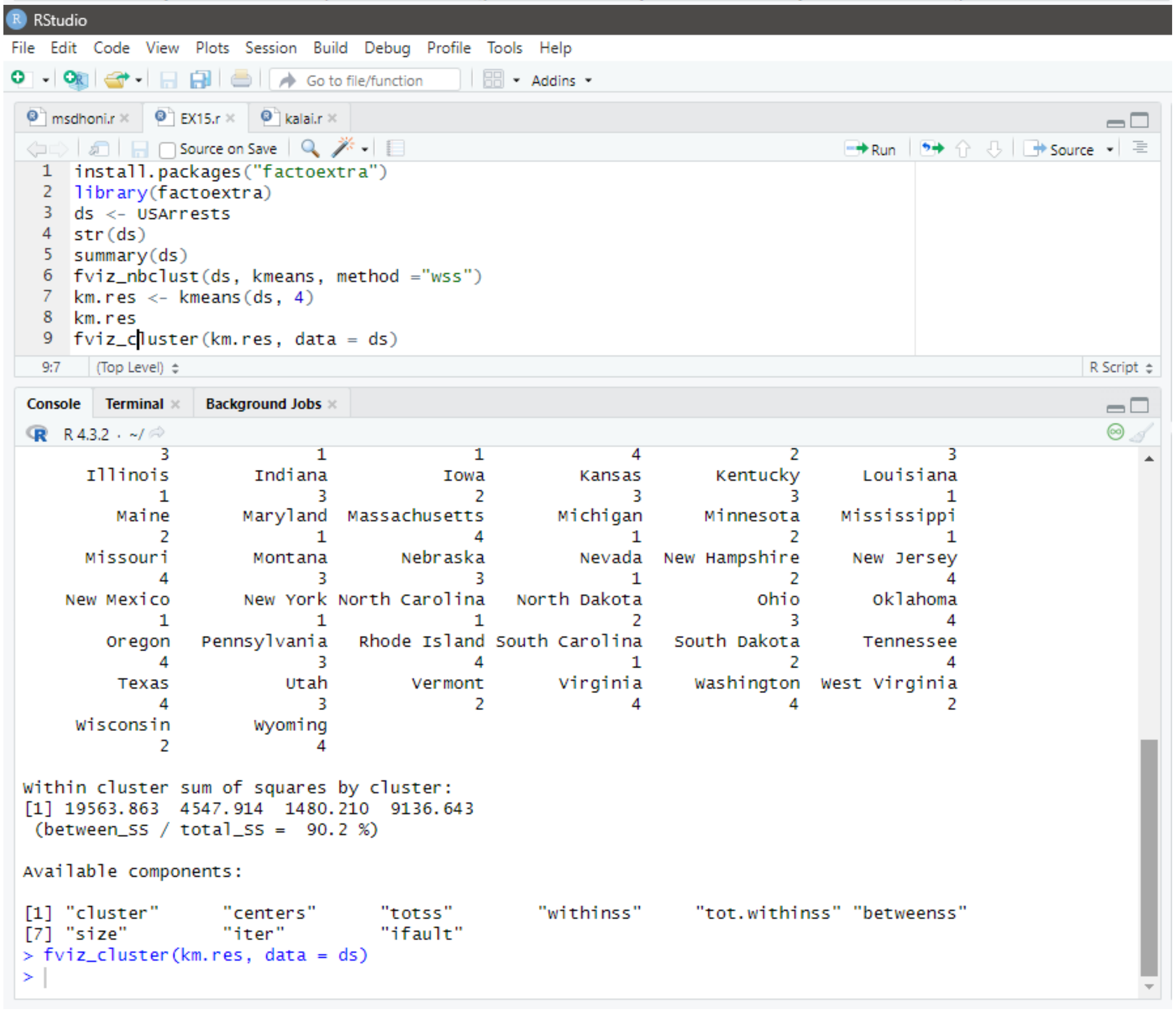
```
install.packages("factoextra")
library(factoextra)
ds <- USArrests
str(ds)
summary(ds)
fviz_nbclust(ds, kmeans, method ="wss")
km.res <- kmeans(ds, 4)
km.res
fviz_cluster(km.res, data = ds)
```

```
> library(factoextra)
> ds <- USArrests
> str(ds)
'data.frame':   50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
> summary(ds)
     Murder          Assault         UrbanPop          Rape      
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30  
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07  
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10  
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23  
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18  
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00  
```



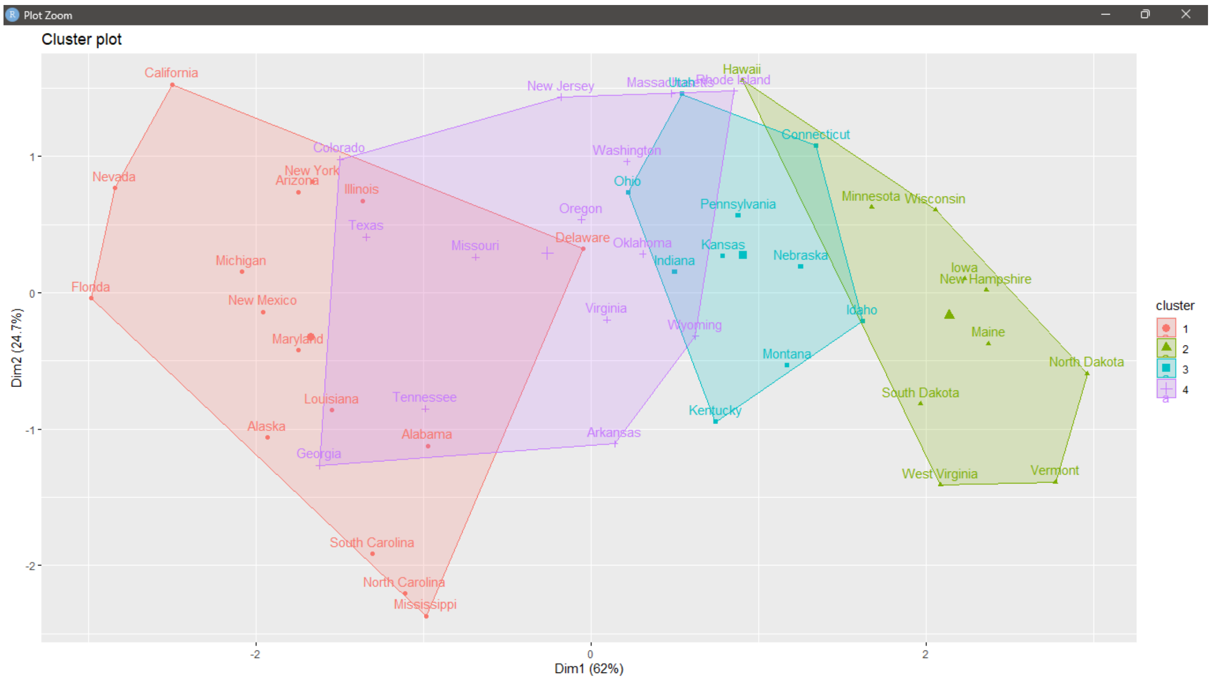
```r
install.packages("factoextra")
library(factoextra)
ds <- USArrests
str(ds)
summary(ds)
fviz_nbclust(ds, kmeans, method ="wss")
km.res <- kmeans(ds, 4)
km.res
fviz_cluster(km.res, data = ds)
```

```
> km.res
K-means clustering with 4 clusters of sizes 16, 10, 10, 14

Cluster means:
     Murder  Assault UrbanPop     Rape
1 11.812500 272.5625 68.31250 28.37500
2  2.950000  62.7000 53.90000 11.51000
3  5.590000 112.4000 65.60000 17.27000
4  8.214286 173.2857 70.64286 22.84286

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



```r
install.packages("factoextra")
library(factoextra)
ds <- USArrests
str(ds)
summary(ds)
fviz_nbclust(ds, kmeans, method ="wss")
km.res <- kmeans(ds, 4)
km.res
fviz_cluster(km.res, data = ds)
```

```
             3              1              1              4              2              3 
      Illinois        Indiana           Iowa         Kansas       Kentucky      Louisiana 
             1              3              2              3              3              1 
         Maine       Maryland  Massachusetts       Michigan      Minnesota    Mississippi 
             2              1              4              1              2              1 
      Missouri        Montana       Nebraska         Nevada  New Hampshire     New Jersey 
             4              3              3              1              2              4 
    New Mexico       New York North Carolina   North Dakota           Ohio       Oklahoma 
             1              1              1              2              3              4 
        Oregon   Pennsylvania   Rhode Island South Carolina   South Dakota      Tennessee 
             4              3              4              1              2              4 
         Texas           Utah        Vermont       Virginia     Washington  West Virginia 
             4              3              2              4              4              2 
     Wisconsin        Wyoming 
             2              4 

Within cluster sum of squares by cluster:
[1] 19563.863  4547.914  1480.210  9136.643
 (between_SS / total_SS =  90.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"   
[7] "size"         "iter"         "ifault"      
> fviz_cluster(km.res, data = ds)
> 
```


Plot Zoom
Cluster plot
California
Nevada
Colorado
New York
Arizona
Illinois
Texas
Michigan
Florida
New Mexico
Maryland
Louisiana
Tennessee
Alaska
Alabama
Georgia
South Carolina
North Carolina
Mississippi
Missouri
Delaware
New Jersey
Massachusetts
Utah
Rhode Island
Hawaii
Washington
Ohio
Oregon
Oklahoma
Indiana
Kansas
Pennsylvania
Connecticut
Nebraska
Virginia
Wyoming
Montana
Kentucky
Arkansas
Idaho
Minnesota
Wisconsin
Iowa
New Hampshire
Maine
North Dakota
South Dakota
West Virginia
Vermont
Dim1 (62%)
Dim2 (24.7%)
cluster
1
2
3
4

EX.NO : 16

DATE :

CLUSTERING USARRESTS USING AGGLOMERATIVE HIERARCHICAL CLUSTERING

AIM :

To show the implementation of Agglomerative Hierarchical Clustering using USArrest Dataset.

AGGLOMERATIVE HIERARCHICAL CLUSTERING :

The Agglomerative Hierarchical Clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's a “bottom-up” approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

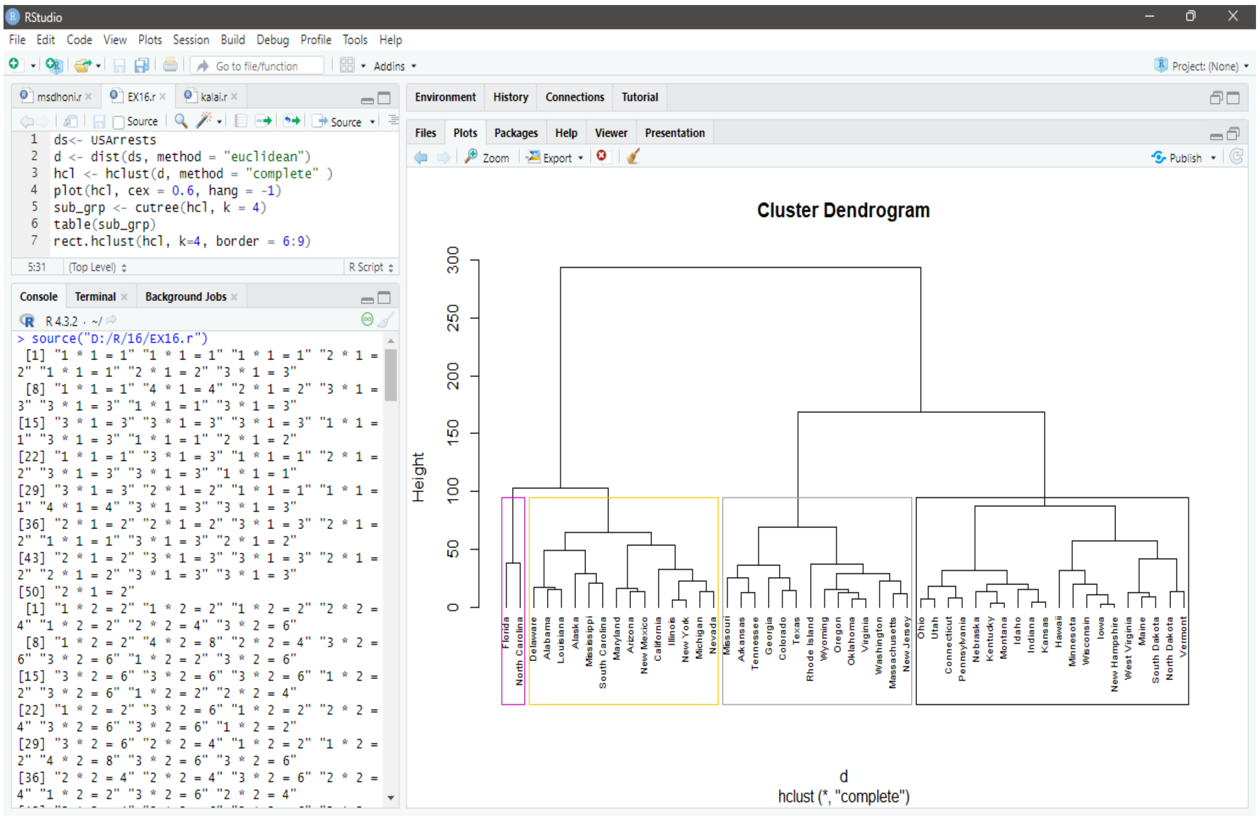
PROCEDURE :

1) Make each data point a single-point cluster → forms N clusters.
2) Take the two closest clusters and make them one cluster → Forms N-1 clusters.
3) Repeat step-3 until you are left with only one cluster.

PROGRAM :

```
ds<- USArrests

d <- dist(ds, method = "euclidean")

hcl <- hclust(d, method = "complete" )

plot(hcl, cex = 0.6, hang = -1)

sub_grp <- cutree(hcl, k = 4)

table(sub_grp)

rect.hclust(hcl, k=4, border = 6:9)
```

OUTPUT :



```r
ds<- USArrests
d <- dist(ds, method = "euclidean")
hcl <- hclust(d, method = "complete" )
plot(hcl, cex = 0.6, hang = -1)
sub_grp <- cutree(hcl, k = 4)
table(sub_grp)
rect.hclust(hcl, k=4, border = 6:9)
```

```
> source("D:/R/16/EX16.r")
 [1] "1 * 1 = 1" "1 * 1 = 1" "1 * 1 = 1" "2 * 1 =
2" "1 * 1 = 1" "2 * 1 = 2" "3 * 1 = 3"
 [8] "1 * 1 = 1" "4 * 1 = 4" "2 * 1 = 2" "3 * 1 =
3" "3 * 1 = 3" "1 * 1 = 1" "3 * 1 = 3"
[15] "3 * 1 = 3" "3 * 1 = 3" "3 * 1 = 3" "1 * 1 =
1" "3 * 1 = 3" "1 * 1 = 1" "2 * 1 = 2"
[22] "1 * 1 = 1" "3 * 1 = 3" "1 * 1 = 1" "2 * 1 =
2" "3 * 1 = 3" "3 * 1 = 3" "1 * 1 = 1"
[29] "3 * 1 = 3" "2 * 1 = 2" "1 * 1 = 1" "1 * 1 =
1" "4 * 1 = 4" "3 * 1 = 3" "3 * 1 = 3"
[36] "2 * 1 = 2" "2 * 1 = 2" "3 * 1 = 3" "2 * 1 =
2" "1 * 1 = 1" "3 * 1 = 3" "2 * 1 = 2"
[43] "2 * 1 = 2" "3 * 1 = 3" "3 * 1 = 3" "2 * 1 =
2" "2 * 1 = 2" "3 * 1 = 3" "3 * 1 = 3"
[50] "2 * 1 = 2"
 [1] "1 * 2 = 2" "1 * 2 = 2" "1 * 2 = 2" "2 * 2 =
4" "1 * 2 = 2" "2 * 2 = 4" "3 * 2 = 6"
 [8] "1 * 2 = 2" "4 * 2 = 8" "2 * 2 = 4" "3 * 2 =
6" "3 * 2 = 6" "1 * 2 = 2" "3 * 2 = 6"
[15] "3 * 2 = 6" "3 * 2 = 6" "3 * 2 = 6" "1 * 2 =
2" "3 * 2 = 6" "1 * 2 = 2" "2 * 2 = 4"
[22] "1 * 2 = 2" "3 * 2 = 6" "1 * 2 = 2" "2 * 2 =
4" "3 * 2 = 6" "3 * 2 = 6" "1 * 2 = 2"
[29] "3 * 2 = 6" "2 * 2 = 4" "1 * 2 = 2" "1 * 2 =
2" "4 * 2 = 8" "3 * 2 = 6" "3 * 2 = 6"
[36] "2 * 2 = 4" "2 * 2 = 4" "3 * 2 = 6" "2 * 2 =
4" "1 * 2 = 2" "3 * 2 = 6" "2 * 2 = 4"
```

EX.NO : 17

DATE :

CLUSTERING USARRESTS USING DIVISIVE HIERARCHICAL CLUSTERING

AIM :

To show the implementation of Divisive Hierarchical Clustering using USArrest Dataset.

DIVISIVE HIERARCHICAL CLUSTERING :

Divisive Hierarchical Clustering is a top-down clustering method where we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters.This is exactly opposite to Agglomerative clustering.Finally, we proceed recursively on each cluster until there is one cluster for each observation.

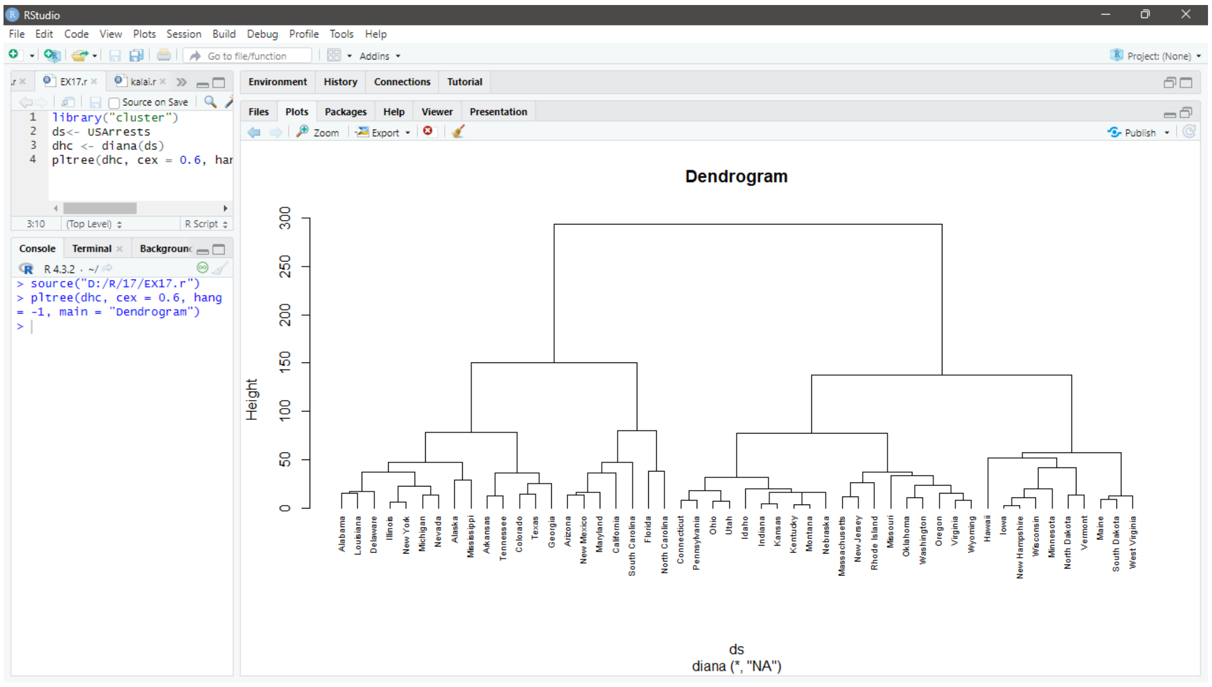
PROCEDURE :

1) Make each data point a single-point cluster → forms N clusters.
2) Take the two closest clusters and make them one cluster → Forms N-1 clusters.
3) Repeat step-3 until you are left with only one cluster.

PROGRAM :

```
library("cluster")

ds<- USArrests

dhc <- diana(ds)

pltree(dhc, cex = 0.6, hang = -1, main = "Dendrogram")
```

OUTPUT :



```
library("cluster")
ds<- USArrests
dhc <- diana(ds)
pltree(dhc, cex = 0.6, har
```

```
R 4.3.2 · ~/
> source("D:/R/17/EX17.r")
> pltree(dhc, cex = 0.6, hang
= -1, main = "Dendrogram")
> |
```

EX.NO : 18

DATE :

CLUSTERING IRIS DATASET USING DBSCAN

AIM :

To show the implementation of DBSCAN using Iris dataset.

DBSCAN :

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density.It groups ‘densely grouped’ data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points. DBSCAN requires only two parameters: epsilon and minPoints. Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

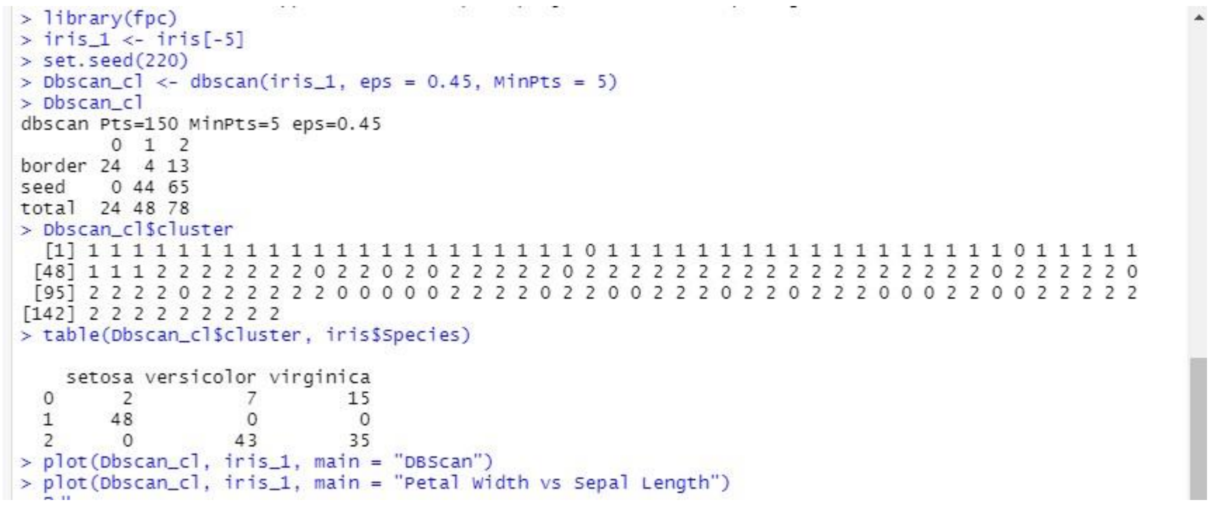
PROCEDURE :

1) Load the dataset.
2) Randomly select a point p.
3) Retrieve all the points that are density reachable from p with regard to maximum radius of the neighbourhood(eps) and minimum number of points within eps neighborhood(Min pts).
4) If the number of points in the neighborhood is more than Min pts then p is a core point.
5) For p core points, a cluster is formed. If p is not a core point, then mark it as a noise/outlier and move to the next point.
6) Continue the process until all the points have been processed.

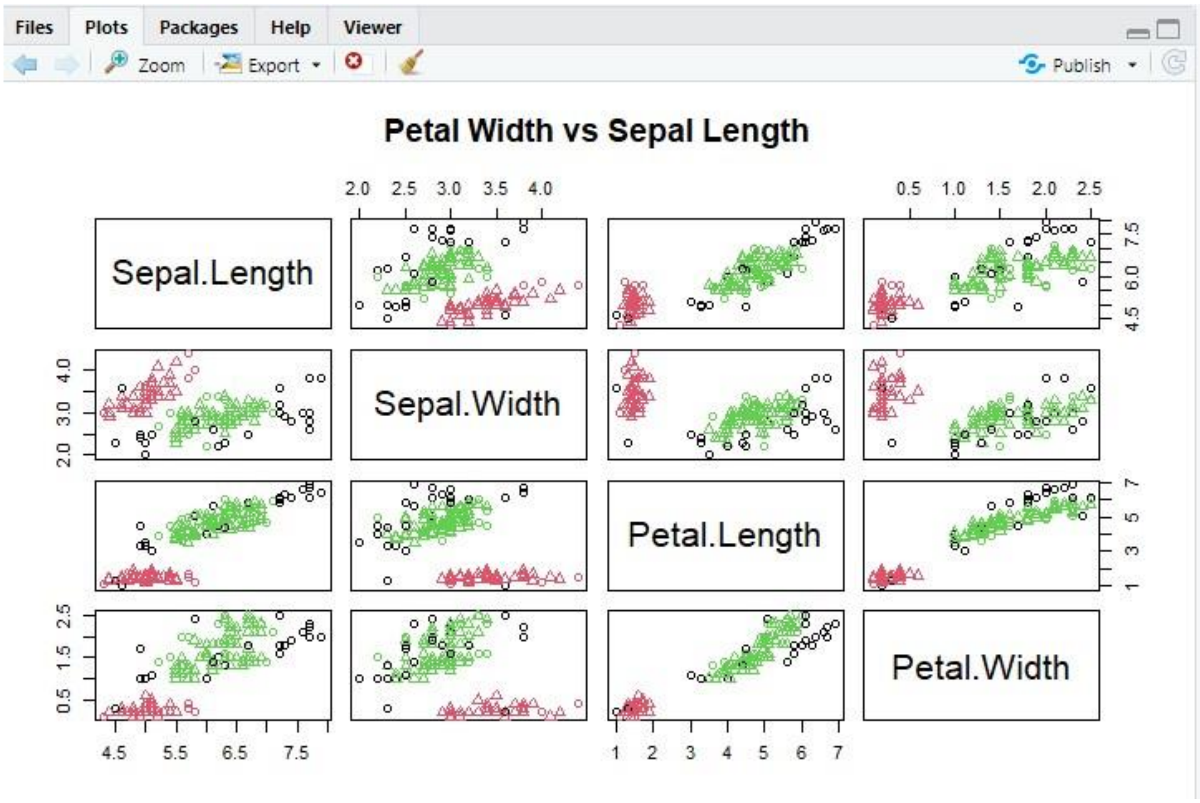
PROGRAM :

```
install.packages("fpc")

library(fpc)

iris_1 <- iris[-5]

set.seed(220)

Dbscan_cl <- dbscan(iris_1, eps = 0.45, MinPts = 5)

Dbscan_cl

Dbscan_cl$cluster

table(Dbscan_cl$cluster, iris$Species)

plot(Dbscan_cl, iris_1, main = "DBScan")

plot(Dbscan_cl, iris_1, main = "Petal Width vs Sepal Length")
```

OUTPUT :



```
> library(fpc)
> iris_1 <- iris[-5]
> set.seed(220)
> Dbscan_cl <- dbscan(iris_1, eps = 0.45, MinPts = 5)
> Dbscan_cl
dbscan Pts=150 MinPts=5 eps=0.45
        0  1  2
border 24  4 13
seed    0 44 65
total  24 48 78
> Dbscan_cl$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 [48] 1 1 1 2 2 2 2 2 2 2 0 2 2 0 2 0 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 0
 [95] 2 2 2 2 0 2 2 2 2 2 2 0 0 0 0 0 2 2 2 2 0 2 2 0 0 2 2 2 0 2 2 0 2 2 2 0 0 0 2 2 0 0 2 2 2 2 2
[142] 2 2 2 2 2 2 2 2 2
> table(Dbscan_cl$cluster, iris$Species)

    setosa versicolor virginica
  0      2          7        15
  1     48          0         0
  2      0         43        35
> plot(Dbscan_cl, iris_1, main = "DBScan")
> plot(Dbscan_cl, iris_1, main = "Petal Width vs Sepal Length")
```


Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Petal Width vs Sepal Length
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 2.5 3.0 3.5 4.0
0.5 1.0 1.5 2.0 2.5
4.5 6.0 7.5
2.0 3.0 4.0
1 3 5 7
0.5 1.5 2.5
4.5 5.5 6.5 7.5
1 2 3 4 5 6 7

EX.NO : 19

DATE :

VISUALIZATIONS

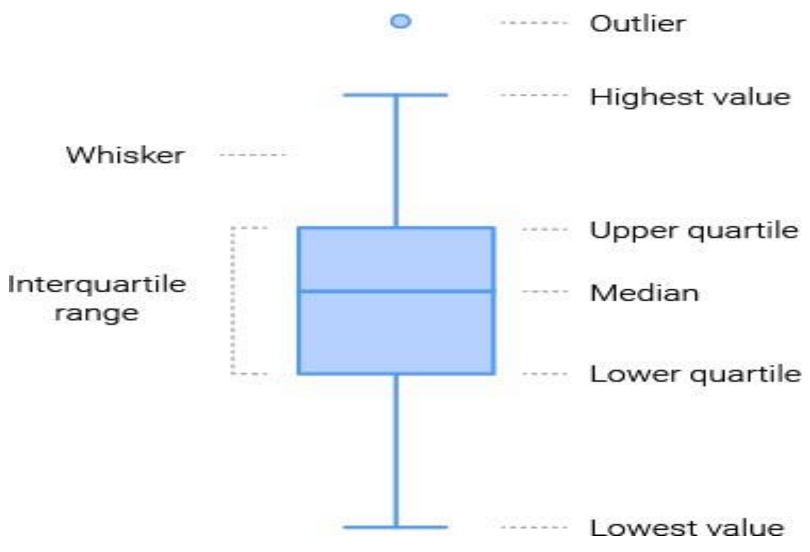
AIM :

To illustrate the concept of Visualization in R.

VISUALIZATIONS :

Data visualization is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

- **Pie Chart :** A Pie Chart is a special chart that shows relative sizes of data using pie slices.
- **Bar Plot :** A bar plot represents data in rectangular bars with length of the bar proportional to the value of the variable.
- **Scatter Plot** : Scatter plots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.
- **Histogram :** A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges.
- **Box Plot :** The box-whisker plot (or a boxplot) is a quick and easy way to visualize complex data where you have multiple samples.



1) Create a pie chart which shows the monthly expenditure?

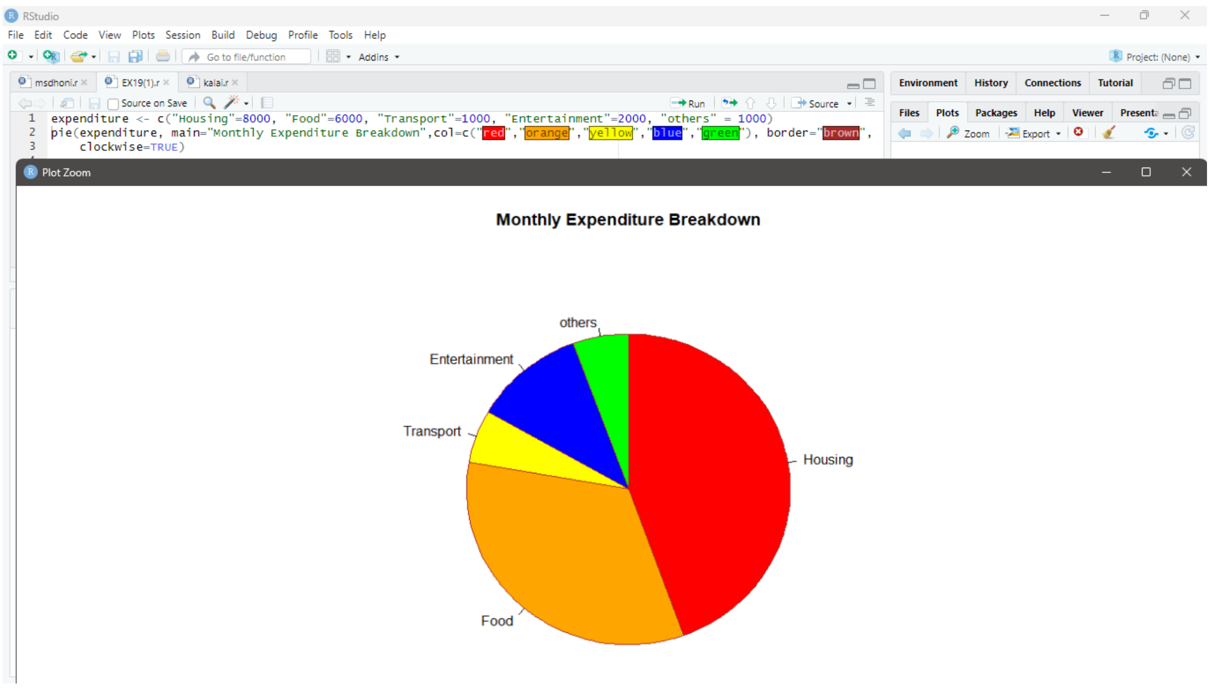
expenditure<-c("Housing"=8000, "Food"=6000, "Transport"=1000, "Entertainment"=2000, "others" = 1000)

#Draw Pie Chart

pie(expenditure, main="Monthly Expenditure Breakdown",

col=c("red","orange","yellow","blue","green"), border="brown", clockwise=TRUE)

OUTPUT :

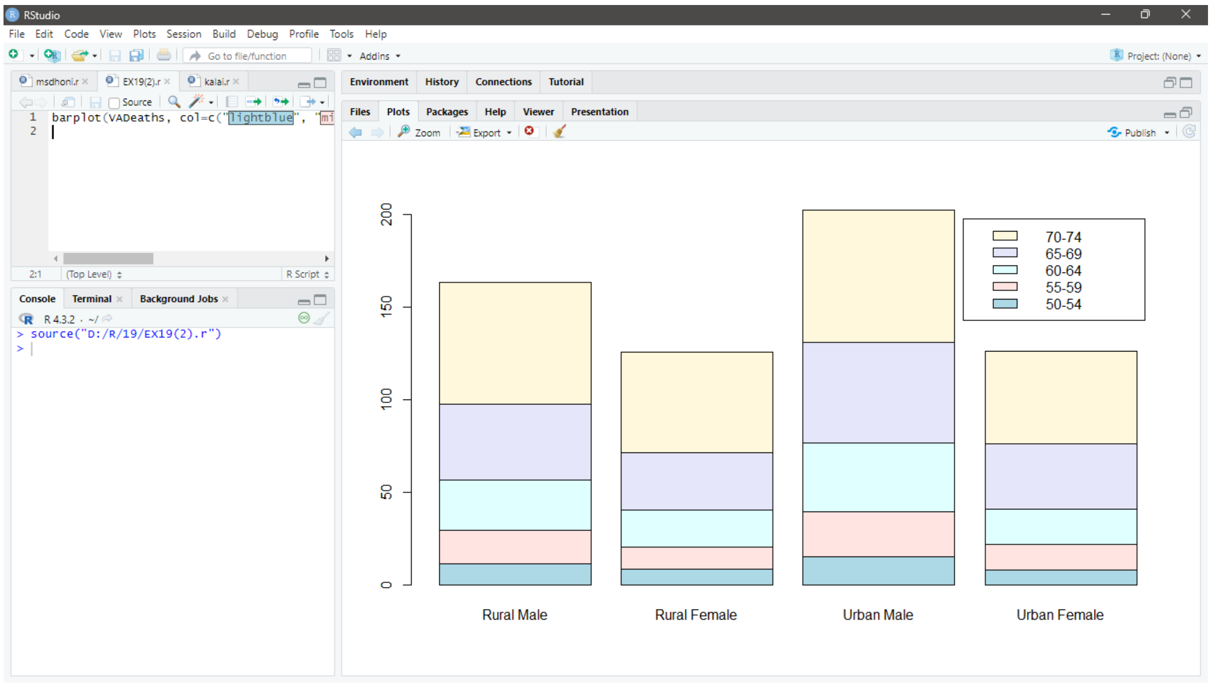


```
expenditure <- c("Housing"=8000, "Food"=6000, "Transport"=1000, "Entertainment"=2000, "others" = 1000)
pie(expenditure, main="Monthly Expenditure Breakdown",col=c("red","orange","yellow","blue","green"), border="brown",
    clockwise=TRUE)
```

2) Create a Bar Plot which will plot the data present in VADeaths dataset.

barplot(VADeaths, col=c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk"), legend=rownames(VADeaths))

OUTPUT :


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
msdhoni.r EX19(2).r kalai.r
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presentation
Zoom Export Publish
200
150
100
50
0
70-74
65-69
60-64
55-59
50-54
Rural Male
Rural Female
Urban Male
Urban Female

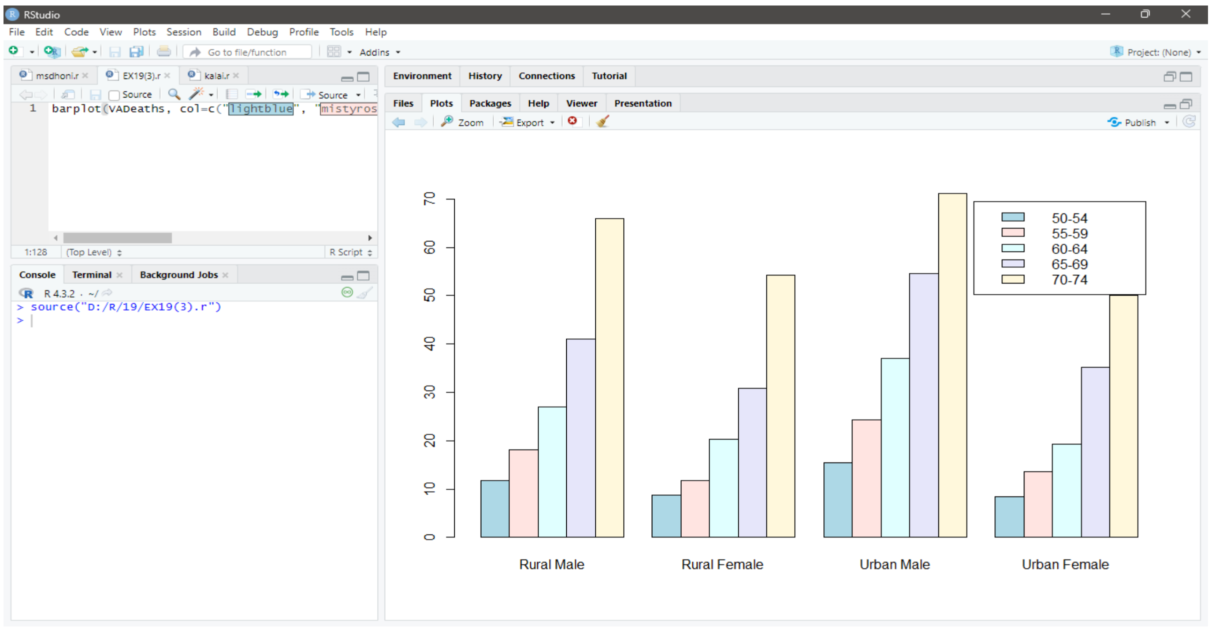
```
barplot(VADeaths, col=c("lightblue", "mi
```

```
R 4.3.2 · ~/
> source("D:/R/19/EX19(2).r")
>
```

3) Create a Bar Plot which will plot the data present in VADeaths dataset.

barplot(VADeaths, col=c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk"), legend=rownames(VADeaths), beside=TRUE)

OUTPUT :


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
msdhoni.r EX19(3).r kalai.r
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presentation
Zoom Export Publish
70
60
50
40
30
20
10
0
50-54
55-59
60-64
65-69
70-74
Rural Male
Rural Female
Urban Male
Urban Female

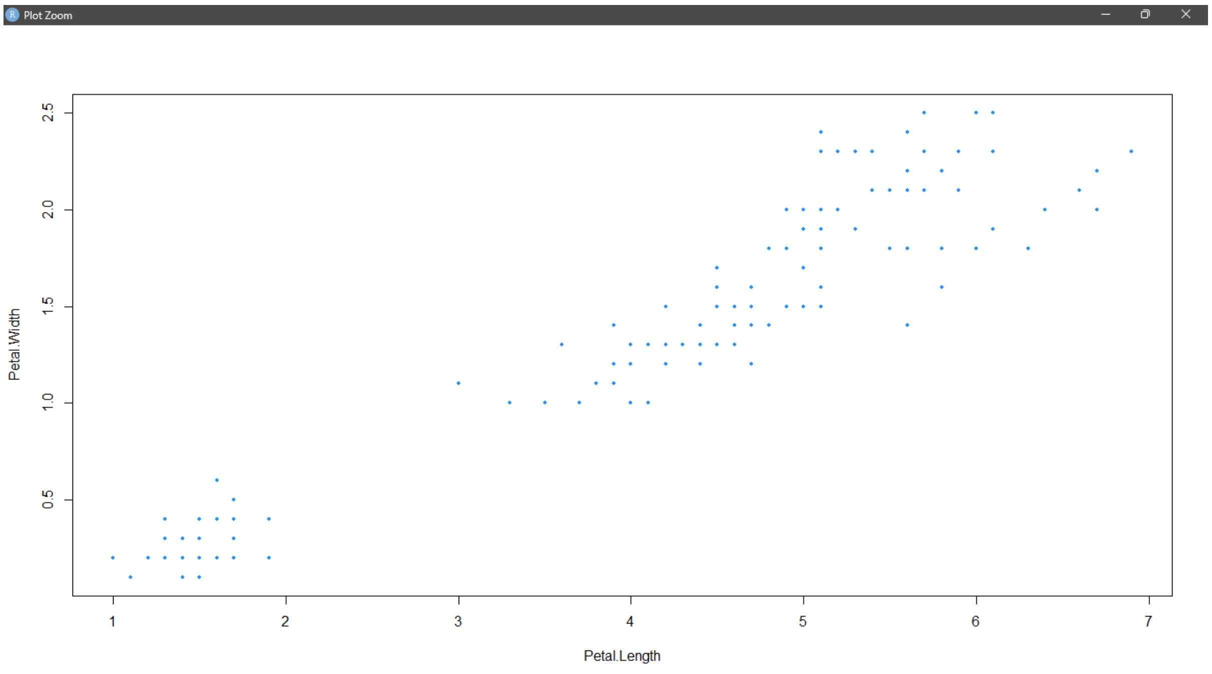
```
barplot(VADeaths, col=c("lightblue", "mistyros
```

```
R 4.3.2 · ~/
> source("D:/R/19/EX19(3).r")
>
```

4) Create a Scatter Plot which will plot the data present in iris dataset.

plot(Petal.Width ~ Petal.Length, data=iris, pch=20, cex=0.8, col="dodgerblue1")

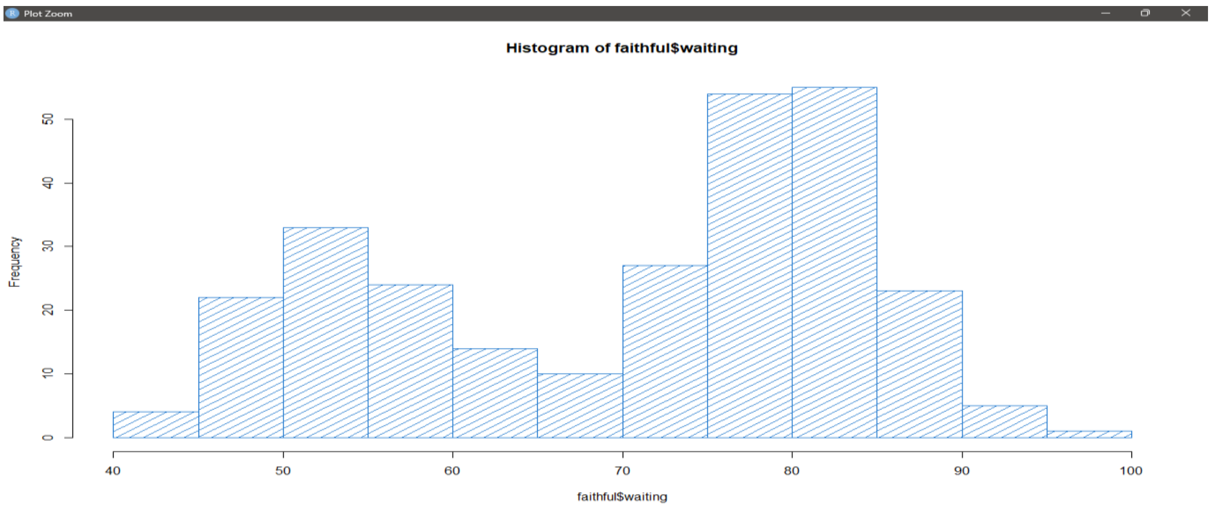
OUTPUT :



5) Create a histogram which will plot the data present in faithful dataset.

hist(faithful$waiting, col="dodgerblue3", density=25, angle=60)

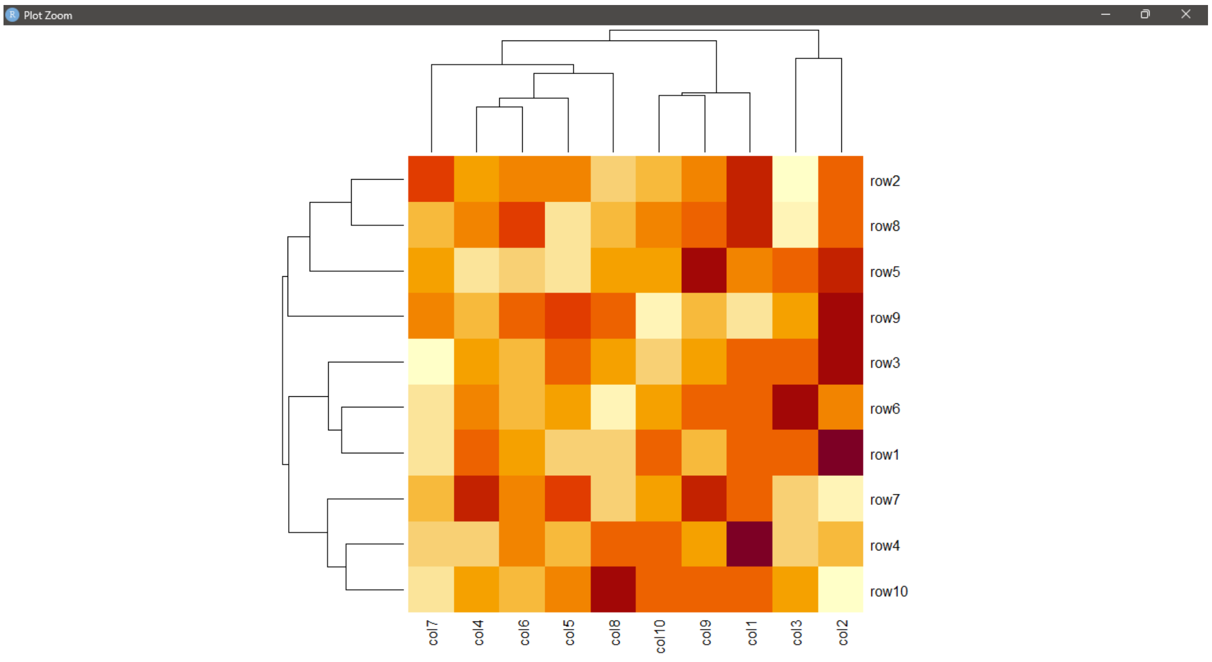
OUTPUT :



6) Create a heatmap in R programming language.

```
set.seed(110)

data <- matrix(rnorm(100, 0, 5), nrow = 10, ncol = 10)

colnames(data) <- paste0("col", 1:10)

rownames(data) <- paste0("row", 1:10)        heatmap(data)
```

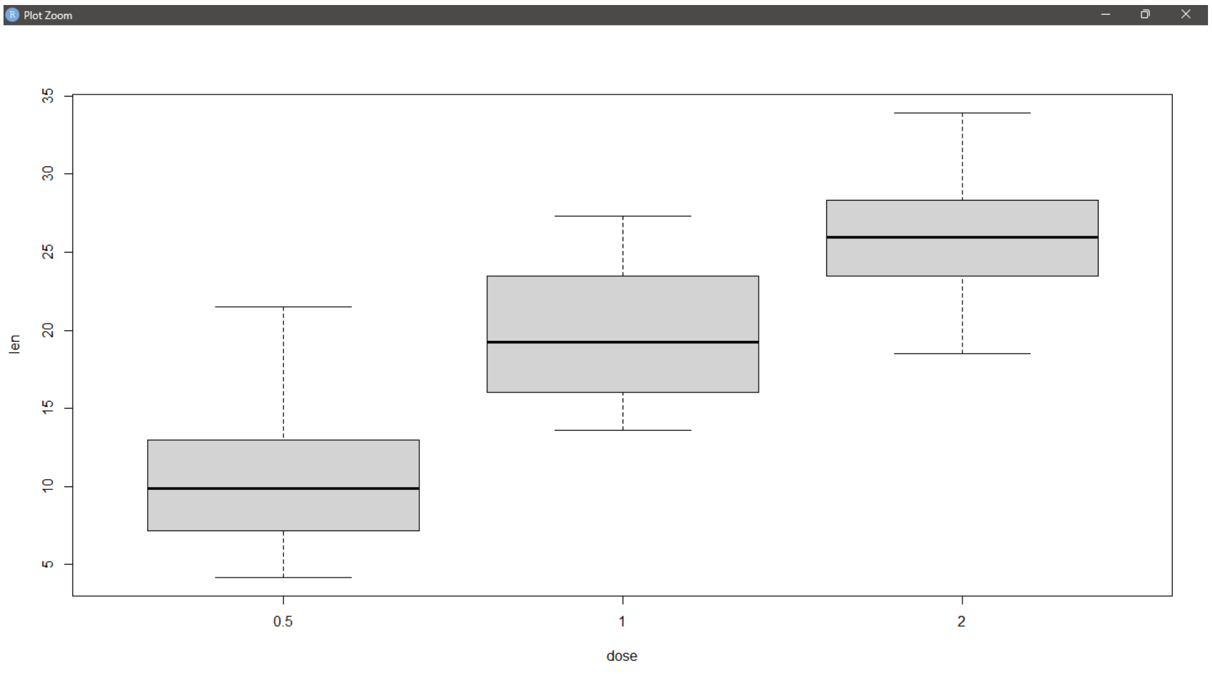
OUTPUT :



7) Create a box plot using R programming language with ToothGrowth dataset.

```
boxplot(len ~ dose, data = ToothGrowth)
```

OUTPUT :


Plot Zoom
len
dose