

LABORATORY EXERCISE BOOK
SUBJECT: APL LAB (CS -215)
B.Tech CSE- 4th SEMESTER

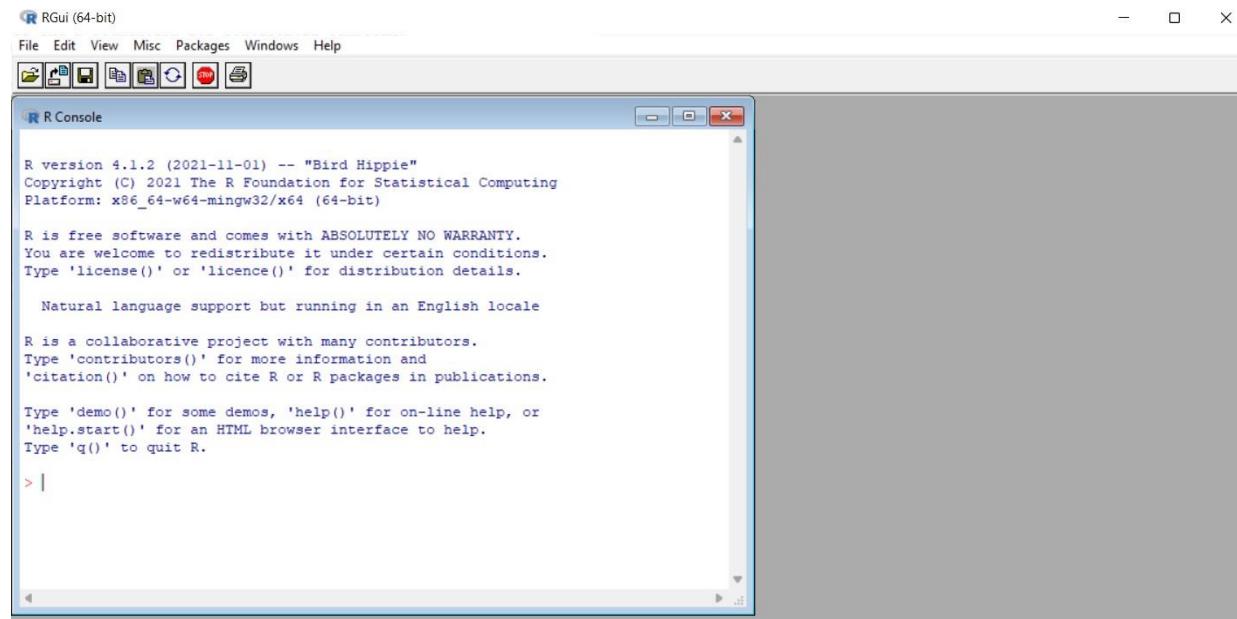


Submitted By: Tushar Rathi
Roll No: 2012174
Branch and Section: CSE [Section-B]

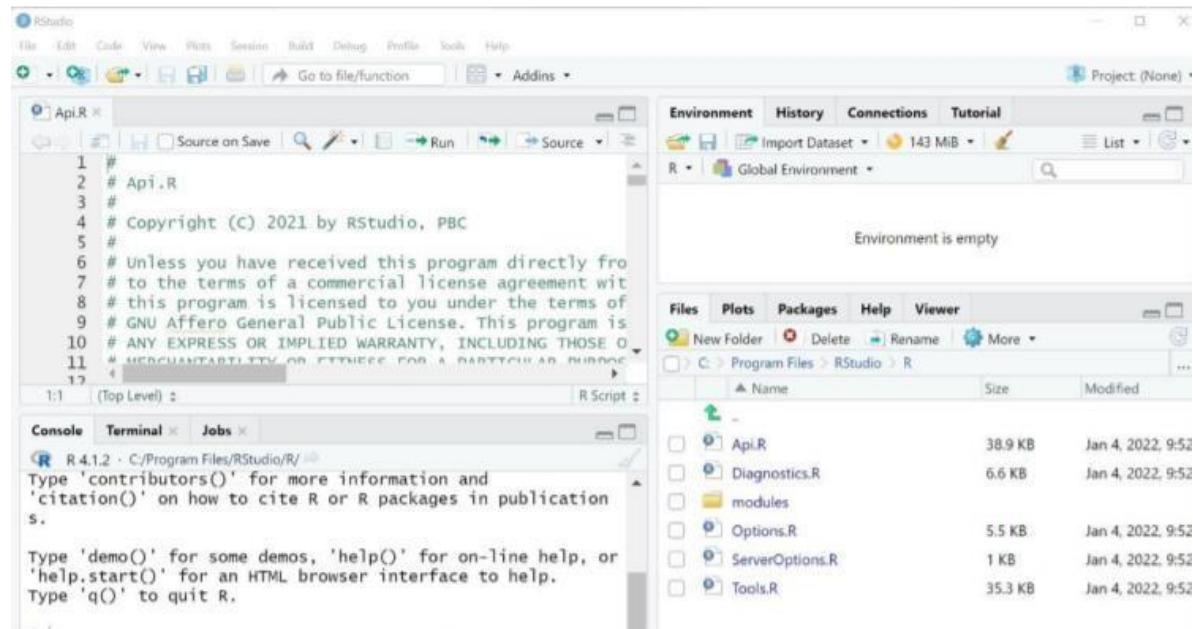
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
ASSAM, PIN-788010

LAB ASSIGNMENT A1:

Task 1: INSTALL R AND R STUDIO



TASK 2: GET FAMILIAR WITH R STUDIO



TASK 3: TRY SOME EXAMPLES OF R COMMANDS

The screenshot shows the RStudio interface with a script file 'Api.R' open. The code in the script is:

```
1 #  
2 # Api.R  
3 <--
```

In the console, the user runs the script and performs some arithmetic operations:

```
> 10^2+36  
[1] 136  
> a=4  
> a  
[1] 4  
> a^5  
[1] 20  
> a=a+10  
> a  
[1] 14  
>
```

The environment pane shows the variable 'a' has a value of 14.

The packages pane lists several R packages:

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-5
geometry	Mesh Generation and Surface Tessellation	0.4.5
linprog	Linear Programming / Optimization	0.9-2
lpSolve	Interface to 'lp_solve' v. 5.5 to Solve Linear/Integer Programs	5.6.15
magic	Create and Investigate Magic Squares	1.5-9
Rcpp	Seamless R and C++ Integration	1.0.8
RcppProgress	An Interruptible Progress Bar with	0.4.2

TASK 4: COMPUTING THE PERCENTAGE OF MY LIFE SPENT AT THE INSTITUTE

The screenshot shows the RStudio interface with a script file 'FirstScript.R' open. The code in the script is:

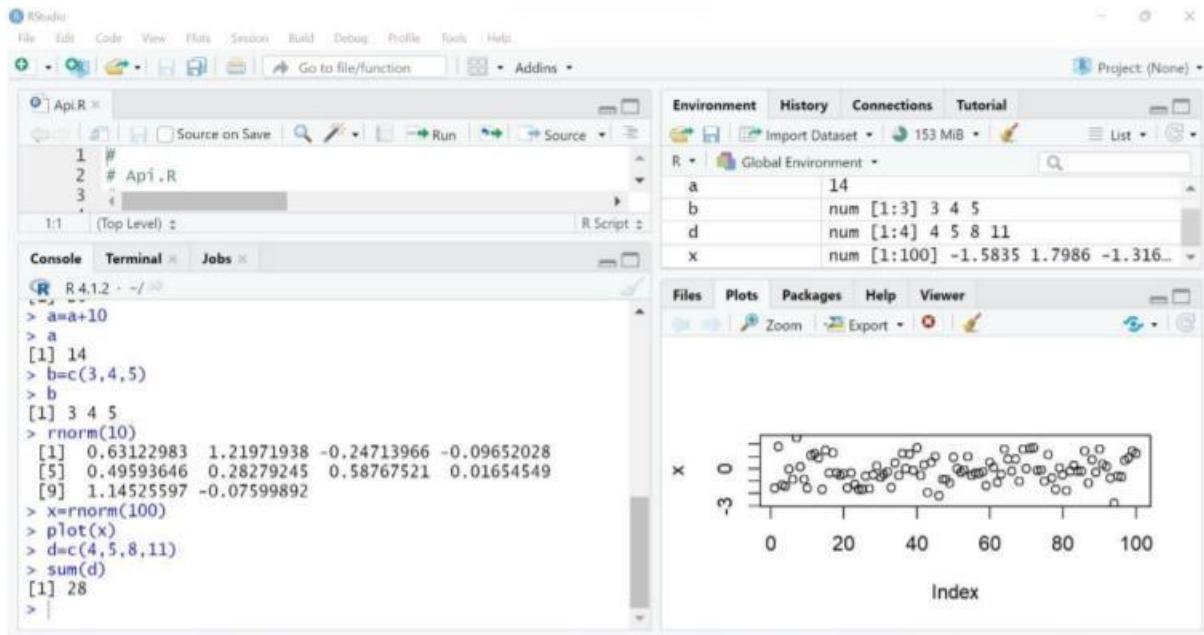
```
1 print("Hello, world!")  
3.6 (Top Level) #
```

In the console, the user runs the script and performs some arithmetic operations:

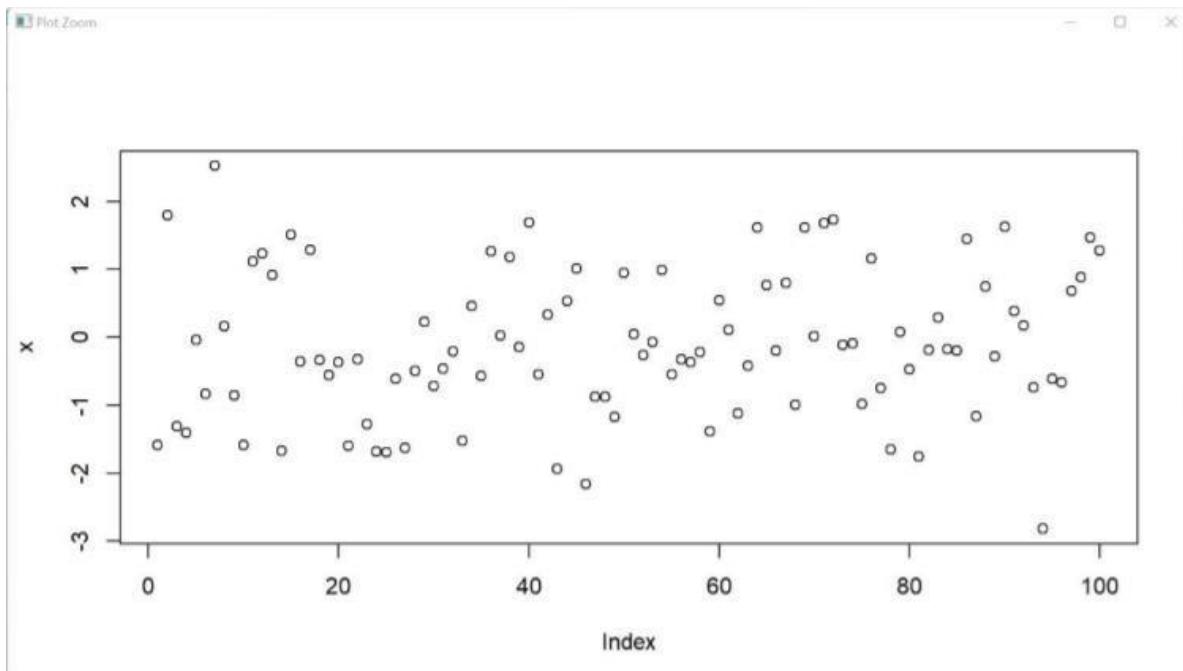
```
Type 'license()' or 'licence()' for distribution details.  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
[workspace loaded from ~/.RData]  
  
> current year= 2022  
Error: unexpected symbol in "current year"  
> currentyear=2022  
> (currentyear-2020)/(currentyear-2002)*100  
[1] 10  
>
```

The environment pane shows the variable 'currentyear' has a value of 2022.

TASK 5: COMPUTING THE SUM OF 4, 5, 8 AND 11 BY COMBINING INTO VECTOR AND USING SUM() FUNCTION



TASK 6: PLOTTING 100 RANDOM NUMBERS



TASK 7: USING SQRT() FUNCTION

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

```
# Api.R
# Api.R
d
```

In the bottom-left pane, the console shows:

```
R 4.1.2 · ~/ ...
> b
[1] 3 4 5
> rnorm(10)
[1] 0.63122983 1.21971938 -0.24713966 -0.09652028
[5] 0.49593646 0.28279245 0.58767521 0.01654549
[9] 1.14525597 -0.07599892
> x=rnorm(100)
> plot(x)
> d=c(4,5,8,11)
> sum(d)
[1] 28
> (2022-2020)/(2022
[1] 10
> sqrt(16)
[1] 4
> sqrt
```

A yellow tooltip is displayed over the 'sqrt' command, providing the following information:

abs(x) computes the absolute value of x, sqrt(x) computes the (principal) square root of x, \sqrt{x} .
The naming follows the standard for computer languages such as C or Fortran.
Press F1 for additional help.

In the top-right pane, the Global Environment shows variables 'a', 'b', 'd', and 'x'. In the bottom-right pane, a scatter plot titled 'Index' is shown with data points ranging from approximately -1.5 to 1.8 across 100 indices.

TASK 8: FILE CONTAINING R CODE THAT GENERATES 100 RANDOM NUMBERS AND PLOT THEM (FIRSTSCRIPT.R)

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

```
x=rnorm(100)
plot(x)
```

In the bottom-left pane, the console shows:

```
R 4.1.2 · ~/ ...
> x= rnorm(100)
> plot(x)
```

In the top-right pane, the Global Environment shows variables 'currentyear' and 'x'. In the bottom-right pane, a scatter plot titled 'Index' is shown with data points ranging from approximately -0.2 to 2.0 across 100 indices.

LAB ASSIGNMENT 2:

Q1: Short notes and sample R code snippet for the following constructs

- i. Vectors
- ii. Lists
- iii. Matrices and Data frames
- iv. Functions definition and function call
- v. Associative arrays
- vi. R data object

Ans:

- i. Vectors

A vector is a data structure that stores a sequence of data elements of the same basic data types. A vector can have a single element or a sequence of elements that belongs to any of the basic data types like logical, integer, Numeric, etc. Therefore vector data structures are further classified into six types. They are:

Data type	Description	Example
Logical	Takes either a TRUE or FALSE value.	TRUE or FALSE
Integer	Take whole number values positive integers and 0	0, 56,9822,100
Double	Takes decimal numbers	5.67, 2.18
Complex	Takes values with real & imaginary parts.	1+2i, -3+4i
Character	Takes a single character or sequence of words	"A", "Hello"
Raw	We can convert character value to its raw value using "charToRaw()" function.	charToRaw("val")

The function c() function is used to combine a sequence of data elements of the same basic data types in R.

Syntax to create vector= c(<value 1>,<value 2>... <value n>)

Arithmetic operations like addition, subtraction, multiplication and division on the vectors are done element wise.

- ii. Lists

Lists are the R objects which contain elements of different types like – numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements. List is created using list() function.

Creating list=

```
list_data <- list(c("Red", "Green"), c(21,32,11), TRUE, c(51.23, 119.1))
```

We can name the elements in the list using this syntax names(list_data) <- c("colours", "numbers", "signals", "decimals")

We can add, delete and update list elements as shown below. We can add and delete elements only at the end of a list. But we can update any element.

iii. Matrices and Data frames

Matrices:

A matrix is a two-dimensional, homogeneous data structure in R. This means that it has two dimensions, rows and columns. A matrix can store data of a single basic type (numeric, logical, character, etc.). Therefore, a matrix can be a combination of two or more vectors.

We can create matrices using the `matrix()` function. The syntax of the `matrix()` function is= `matrix(data, byrow, nrow, ncol, dimnames)`

The arguments in the `matrix` function are the following:

1. data – data contains the elements in the R matrix.
2. byrow – byrow is a logical variable. Matrices are by default column-wise. By setting `byrow` as TRUE, we can arrange the data row-wise in the matrix.
3. `nrow` – defines the number of rows in the R matrix.

4. `ncol` – defines the number of columns in the R matrix.
5. `dimnames` – takes two character arrays as input for row names and column names. We can also name the rows and columns of an R matrix after its creation by using the `rownames()` or `colnames()` functions

It is not necessary to specify both `nrow` and `ncol`. Only one of the two is required. R calculated the other based on the length of the data.

Data Frame:

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

Following are the characteristics of a data frame.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

Syntax of data frame = `data.frame(df, stringsAsFactors = TRUE)`

`df`: It can be a matrix to convert as a data frame or a collection of variables to join `stringsAsFactors`: Convert string to factor by default

The statistical summary and nature of the data can be obtained by applying `summary()` function.

We can expand data frame using `rbind()` and `cbind()` just like in matrices

iv. Function Definitions and Function Call

Function:

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions. An R function is created by using the keyword `function`. The basic syntax of an R function definition is as follows:

Function body

}

The different parts of a function are –

- Function Name – This is the actual name of the function. It is stored in R environment as an object with this name.
- Arguments – An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.
- Function Body – The function body contains a collection of statements that defines what the function does.
- Return Value – The return value of a function is the last expression in the function body to be evaluated.

R has many in-built functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as user defined functions. Simple examples of in-built functions are seq(), mean(), max(), sum(x) and paste(...) etc.

Function Call:

A function call is a request made by a program or script that performs a predetermined function. Most of the computations carried out in R involve the evaluation of functions. We will also refer to this as function invocation.

Functions are invoked by name with a list of arguments separated by commas.

Example= mean(1:10)

In this example the function mean was called with one argument, the vector of integers from 1 to 10.

Function calls can have tagged (or named) arguments, as in plot(x, y, pch = 3).

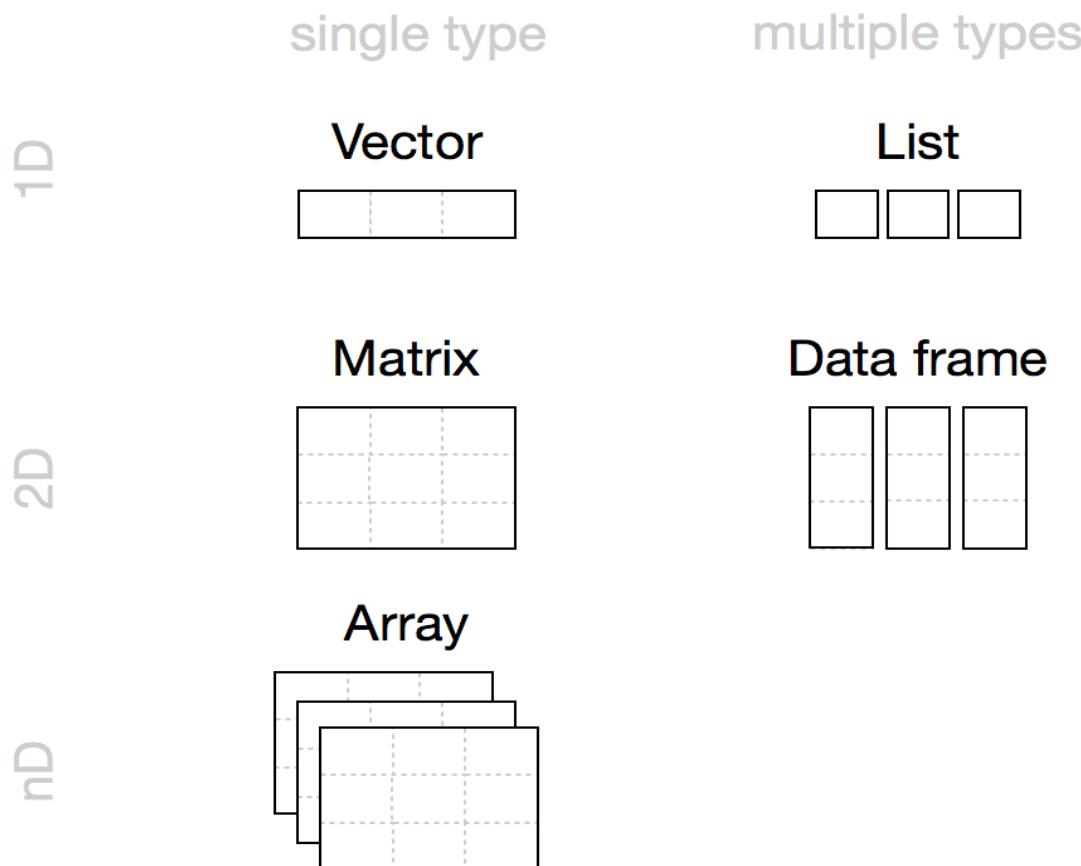
v. Associative arrays

Hash table or associative array, is a well-known key value data structure. To implement the equivalent R, we can use a vector of any type, a list, or an environment.

vi. R data objects

There are 5 basic types of objects in the R language:

- Atomic Vectors: A vector is a data structure that stores a sequence of data elements of the same basic data types.
- Lists: Lists are the R objects which contain elements of different types like – numbers, strings, vectors and another list inside it.
- Matrices: Matrices store values in a two-dimensional array, just like a matrix from linear algebra.
- Arrays: The array function creates an n-dimensional array. You could use array to sort values into a cube of three dimensions or a hypercube in 4, 5, or n dimensions.
- Data Frames: Data frames are the two-dimensional version of a list.



2.1) R Program to add two vectors of integers type.

The screenshot shows the RStudio interface with the following details:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** History, R Objects, Source on Save, Run, Source.
- Script Editor:** Contains the following R code:

```
1 x = c(50, 40)
2 y = c(30, 10)
3 print("Original vectors:")
4 print(x)
5 print(y)
6 print("After adding two vectors:")
7 z = x + y
8 print(z)
9
10 a=c(1,30)
11 b=c(10,20)
12 sum=a+b
13 print(sum)
```
- Environment Tab:** Shows variables **a**, **b**, and **sum** with their corresponding values.
- Console Tab:** Displays the R session output for the provided code.
- Bottom Status Bar:** Shows the date and time (10:12 PM, 2/12/2022), weather (22°C Light rain), and system information (ENG).

2.2) R program to create a list containing strings, numbers, vectors and a logical values.

The screenshot shows the RStudio interface with the following details:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** History, R Objects, Source on Save, Run, Source.
- Script Editor:** Contains the following R code:

```
1 x = c(50, 40)
2 y = c(30, 10)
3 print("Original vectors:")
4 print(x)
5 print(y)
6 print("After adding two vectors:")
7 z = x + y
8 print(z)
9
10 a=c(1,30)
11 b=c(10,20)
12 sum=a+b
13 print(sum)
14
15 myList = list("R", "Neha", c(5, 7, 9, 11), TRUE, 125.17, 75.83)
16 print("Data of the list:")
17 print(myList)
```
- Environment Tab:** Shows a list named **myList** with 6 elements, and individual variable values **a**, **b**, and **sum**.
- Console Tab:** Displays the R session output for the provided code.
- Bottom Status Bar:** Shows the date and time (10:13 PM, 2/12/2022), weather (22°C Light rain), and system information (ENG).

2.3.1) R program to add two matrixes.

The screenshot shows the RStudio interface with the following code in the script editor:

```
7 Z = X + Y
8 print(Z)
9
10 a=c(1,30)
11 b=c(10,20)
12 sum=a+b
13 print(sum)
14
15 myList = list("R", "Neha", c(5, 7, 9, 11), TRUE, 125.17, 75.83)
16 print("Data of the list:")
17 print(myList)
18
19 a=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
20 b=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=TRUE)
21 print(a)
22 print(b)
23 c=a+b
24 print(c)
25
26
```

The console output shows the execution of the code, printing the matrices and their sum:

```
[1] [2] [3]
[1,] 1 2 3
[2,] 4 5 6
> print(b)
[1,] [2] [3]
[1,] 1 2 3
[2,] 4 5 6
> a=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
> b=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=TRUE)
> print(a)
[1,] [2] [3]
[1,] 1 2 3
[2,] 4 5 6
> print(b)
[1,] [2] [3]
[1,] 1 2 3
[2,] 4 5 6
> c=a+b
> print(c)
[1,] [2] [3]
[1,] 2 4 6
[2,] 6 9 12
```

The environment pane shows variables defined in the global environment:

- a: num [1:2, 1:3] 1 2 3 4 5 6
- b: num [1:2, 1:3] 1 4 2 5 3 6
- c: num [1:2, 1:3] 2 6 5 9 8 12
- MyList: List of 6
- MyMatrix: num [1:2, 1:3] 1 2 3 4 5 6
- values: num [1:2] 11 50
- sum: num [1:2] 11 50

2.3.2) R program to extract first two rows from a given data frame.

The screenshot shows the RStudio interface with the following code in the script editor:

```
16 print("Data of the list:")
17 print(myList)
18
19 a=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
20 b=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=TRUE)
21 print(a)
22 print(b)
23 c=a+b
24 print(c)
25
26 Mydataframe=data.frame(
27   name=c("Rahul", "Reem", "Danial"),
28   age=c(20,18,19),
29   gender=c("M", "F", "M")
30 )
31 print(Mydataframe)
32 result=Mydataframe[1:2,]
33 print(result)
34
35
```

The console output shows the creation of a data frame and its first two rows:

```
3 Rahul 20 M
3 Reem 18 F
3 Danial 19 M
> result=Mydataframe[1:2,]
> print(result)
  name age gender
1 Rahul 20     M
2 Reem 18     F
```

The environment pane shows variables defined in the global environment:

- C: num [1:2, 1:3] 2 6 5 9 8 12
- Mydataframe: 3 obs. of 3 variables
 - \$ name : chr "Rahul" "Reem" "Danial"
 - \$ age : num 20 18 19
 - \$ gender: chr "M" "F" "M"
- MyList: List of 6
- MyMatrix: num [1:2, 1:3] 1 2 3 4 5 6
- result: 2 obs. of 3 variables
 - gender: chr "M" "F" "M"
 - sum: num [1:2] 11 50
- values: num [1:2] 11 50

2.4) R program to calculate area and perimeter of a rectangle.

The screenshot shows the RStudio interface with the following details:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** History, ROjects.Rproj, Source, Run, Source, Environment, History, Connections, Tutorial.
- Code Editor:** Contains R code for creating a data frame and a rectangle function.
- Console:** Shows the execution of the R code, including the creation of a data frame named MydataFrame and a rectangle function named Newrectangle.
- Environment:** Shows objects in the global environment: \$age, \$gender, Mylist, MyMatrix, NewRectangle, result, Values, gender, sum, Functions, Rectangle.
- Status Bar:** 22°C Light rain, ENG, 10:56 PM, 2/12/2022.

2.5) Create a hash table or Associative array using different R data structures.

The screenshot shows the RStudio interface with the following details:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Left Panel:** History (with a search bar), Objects.Rnw (selected), Source on Save, Run, Source.
- Code Editor:** Displays R code for generating new strings based on input length and character sets.
- Console:** Shows the R session output, including the creation of a function `unique_strings` and its execution.
- Environment Tab:** Shows the global environment with objects like `age`, `gender`, `myenv`, `mykey`, `mylist`, `MyMatrix`, `NewRectangle`, `result`, and `values`.
- Files Tab:** Shows the current file `Objects.Rnw` is selected.
- Plots Tab:** No plots are currently displayed.
- Packages Tab:** Shows available packages: abind, geometry, linprog, lpSolve, magic, Rcpp, and RcppProgress.
- Help Tab:** Not visible in the screenshot.
- Viewer Tab:** Not visible in the screenshot.
- Bottom Status Bar:** Shows the system tray with icons for network, battery, volume, and system status.

2.6) R data objects:

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays an R script with several examples of data structures and operations.
- Console:** Shows the R session output corresponding to the code in the editor.
- Environment View:** Shows the global environment with variables like `a`, `b`, `c`, `MyDataFrame`, `MyList`, `result`, and `values`.
- File Bar:** Includes tabs for History, Objects, and Source on Save.
- System Tray:** Shows the date (2/12/2022), time (11:25 PM), and weather (22°C Light rain).

```

# Vectors
a=c(1,30)
b=c(10,20)
sum=a+b
print(sum)

# List
myList = list("R", "Neha", c(5, 7, 9, 11), TRUE, 125.17, 75.83)
print("Data of the list:")
print(myList)

# Matrix
a=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3)
b=matrix(c(1,2,3,4,5,6), nrow=2, ncol=3, byrow=TRUE)
print(a)
print(b)
c=a+b
print(c)

# Data Frame
MyDataFrame=data.frame(
  name=c("Rahul", "Reem", "Danial"),
  age=c(20,18,19),
  gender=c('M', 'F', 'M')
)
print(MyDataFrame)
result=MyDataFrame[1:2]
print(result)

# Factor
age=c(20,19,18)
gender=c('M', 'F', 'M')
print(result)
print(result$age)
print(result$gender)

# Arrays
arr = array(2:13, dim = c(2, 3, 2))
print(arr)

```

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays an R script with examples involving factors and arrays.
- Console:** Shows the R session output corresponding to the code in the editor.
- Environment View:** Shows the global environment with variables like `a`, `b`, `c`, `MyDataFrame`, `MyList`, `result`, and `values`.
- File Bar:** Includes tabs for History, Objects, and Source on Save.
- System Tray:** Shows the date (2/12/2022), time (11:29 PM), and weather (22°C Light rain).

```

# Factor
s <- c("spring", "autumn", "winter", "summer",
      "spring", "autumn")
print(factor(s))

# Arrays
# arranges data from 2 to 13
# in two matrices of dimensions 2x3
arr = array(2:13, dim = c(2, 3, 2))
print(arr)

```

APL Assignment 3

Q1: Describe the types of tailed histograms such as long-tailed, short-tailed and skewed-tailed and provide a sample of R code snippets for each type.

Ans:

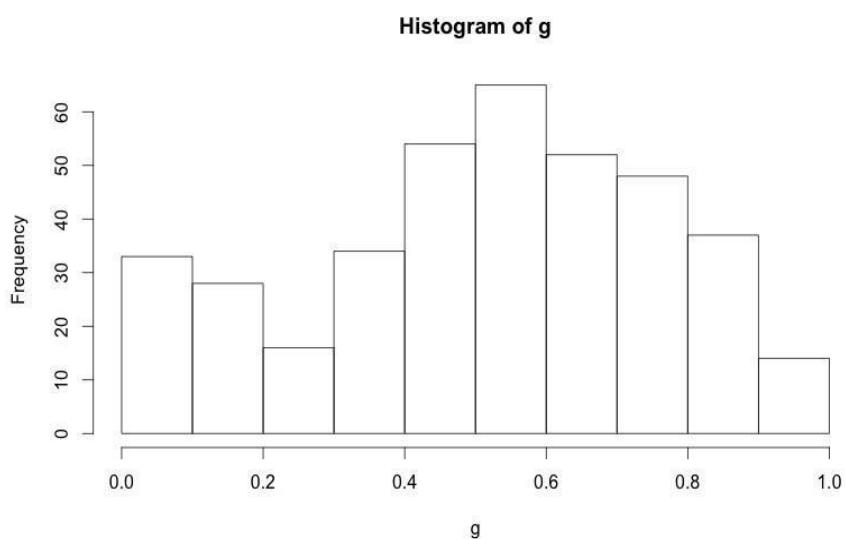
Histogram: A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chart but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range. The most common form of the histogram is obtained by splitting the range of the data into equal-sized bins (called classes).

R creates histogram using `hist()` function. This function takes a vector as an input and uses some more parameters to plot histograms.

The basic syntax for creating a histogram using R is – `hist(v, main, xlab, xlim, ylim, breaks, col, border)`

Following is the description of the parameters used –

- `v` is a vector containing numeric values used in histogram.
- `main` indicates title of the chart.
- `col` is used to set color of the bars.
- `border` is used to set border color of each bar.
- `xlab` is used to give description of x-axis.
- `xlim` is used to specify the range of values on the x-axis.
- `ylim` is used to specify the range of values on the y-axis.
- `breaks` is used to mention the width of each bar.



The histogram graphically shows the following :

- center (i.e. the location) of the data
- spread (i.e. the scale) of the data
- skewness of the data
- presence of outliers; and
- presence of multiple modes in the data.

Types of Histograms with Examples:

□ Short tailed histograms

- For a short-tailed distribution, the tails approach zero very fast.
- Such distributions commonly have a truncated ("sawed-off") look.
- The classical short-tailed distribution is the uniform (rectangular) distribution in which the probability is constant over a given range and then drops to zero everywhere else, we would speak of this as having no tails, or extremely short tails.

Example Code:

```
n = 100
```

```
#short tailed: Uniform (on [0,2]) with mu = 1 and sigma = 0.3333; short <-
```

```
runif(n,min=0,max=2) RandomData <- short
```

```
title <- "Right tailed Distribution"
```

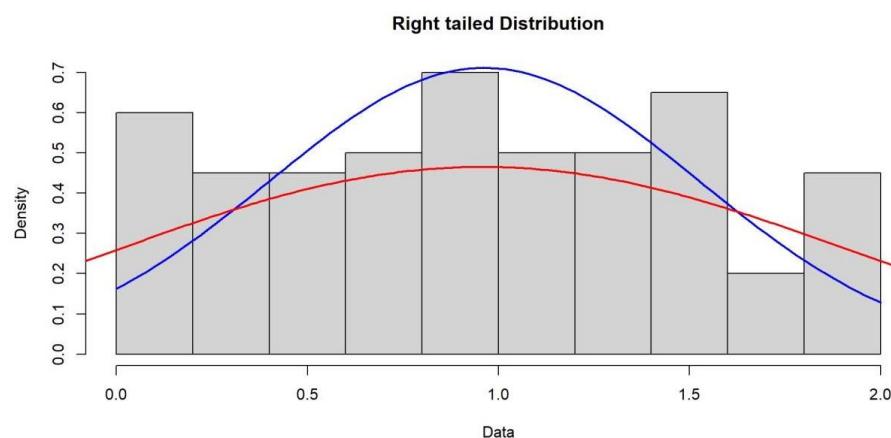
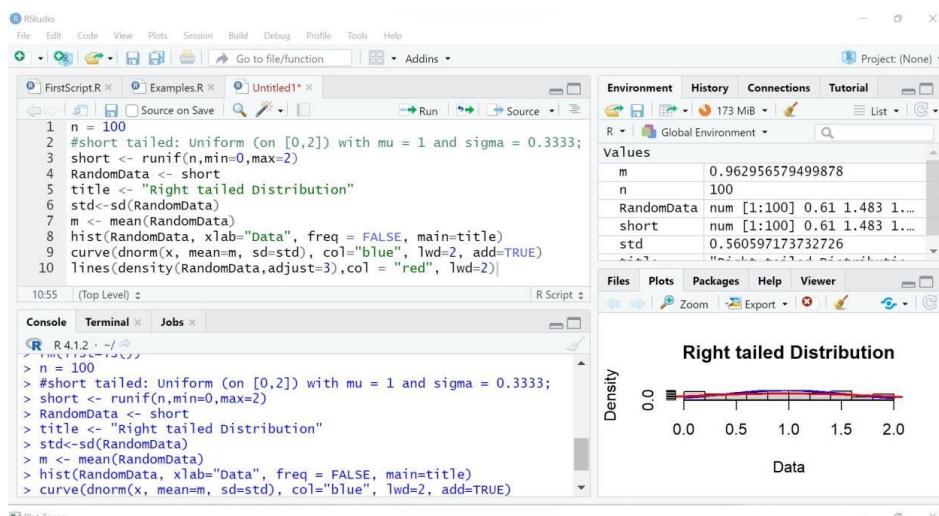
```
std<-sd(RandomData) m <-
```

```
mean(RandomData)
```

```
hist(RandomData, xlab="Data", freq = FALSE, main=title) curve(dnorm(x,
```

```
mean=m, sd=std), col="blue", lwd=2, add=TRUE)
```

```
lines(density(RandomData,adjust=3),col = "red", lwd=2)
```

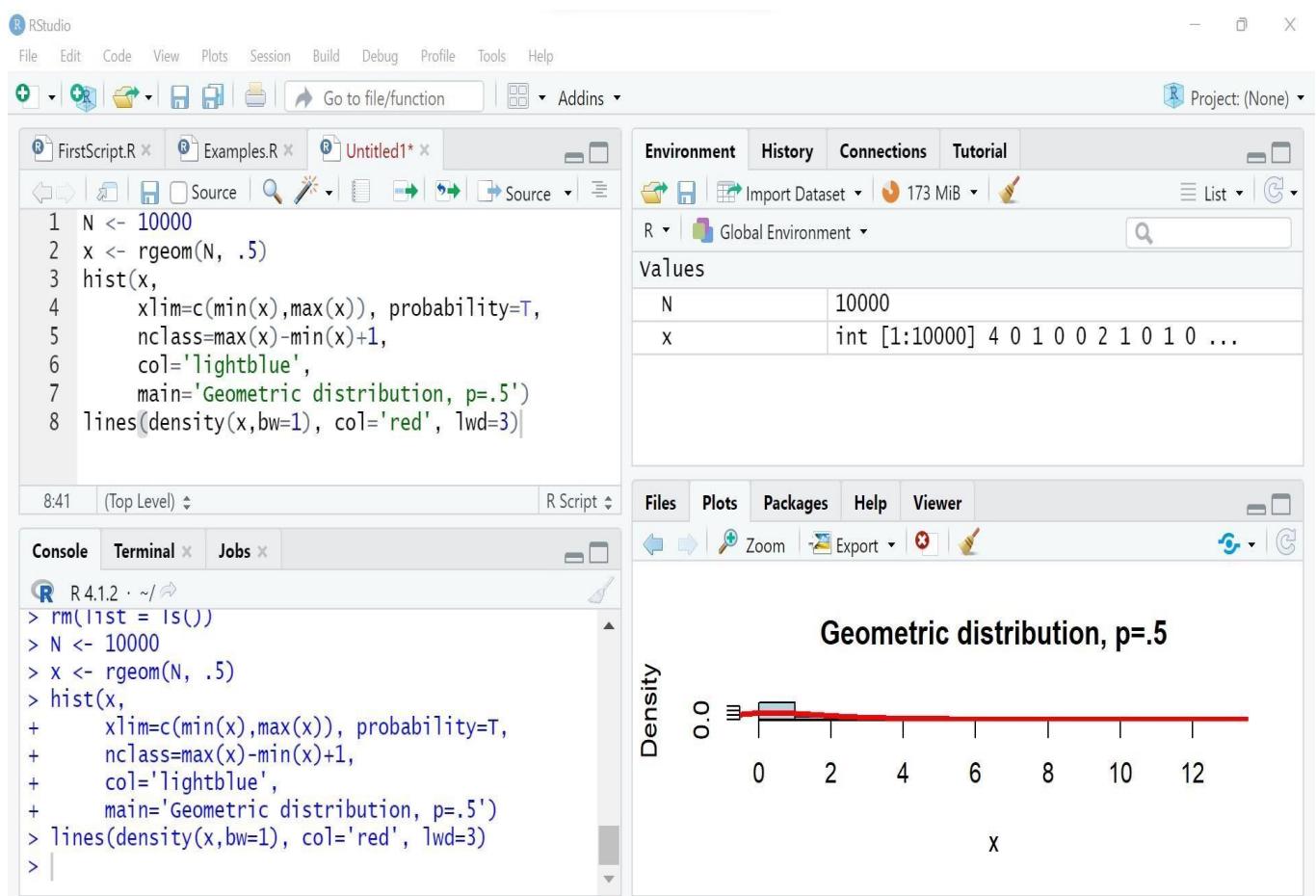


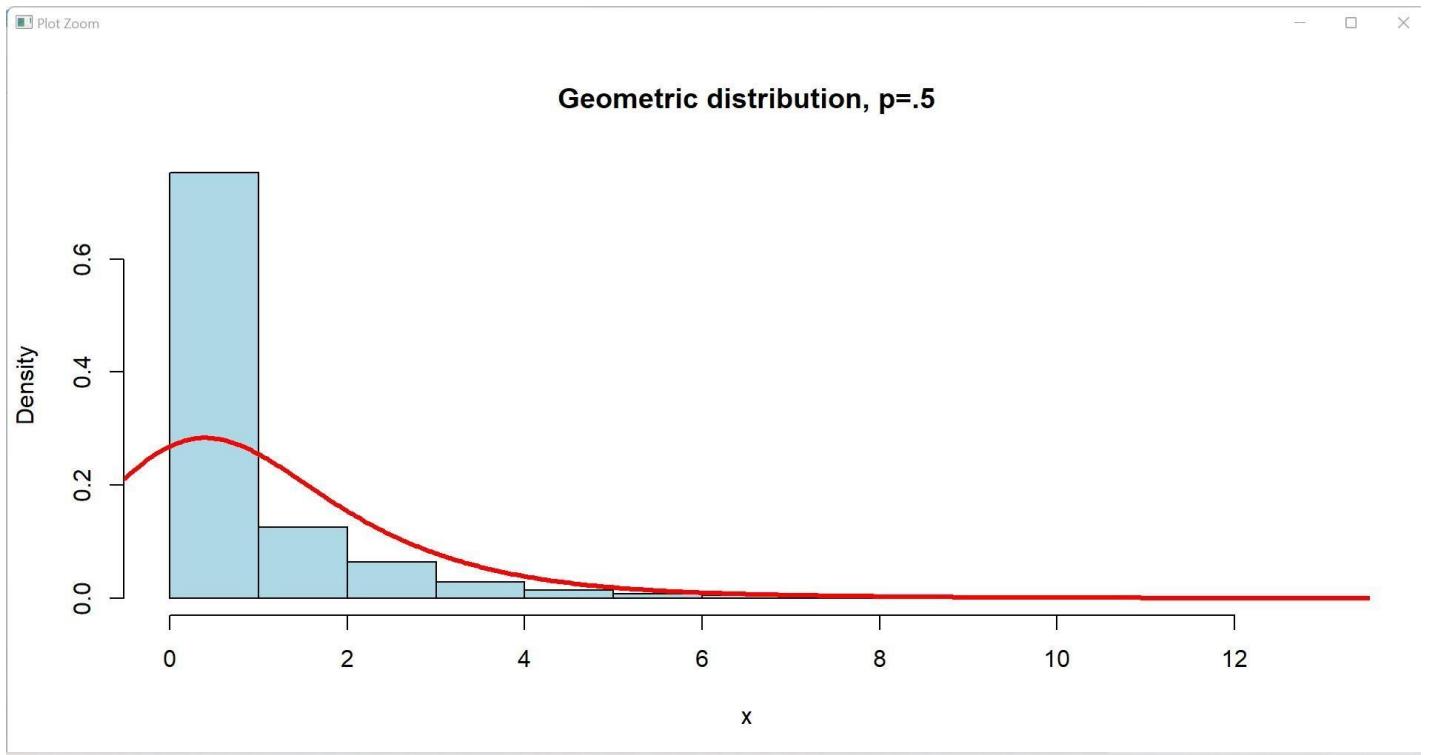
□ Long tailed histograms

- For a long-tailed distribution, the tails decline to zero very slowly, and hence one is apt to see probability a long way from the body of the distribution. The classical long-tailed distribution is the Cauchy distribution.

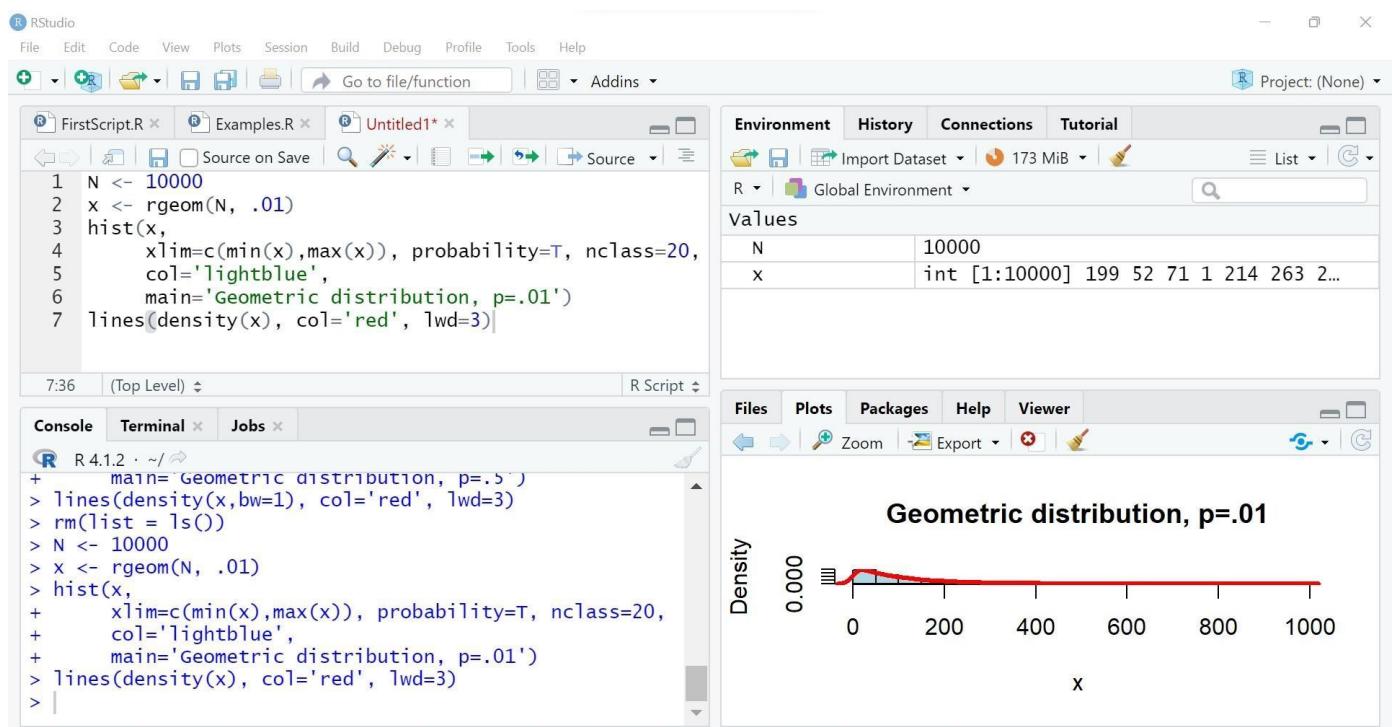
Example Code 1:

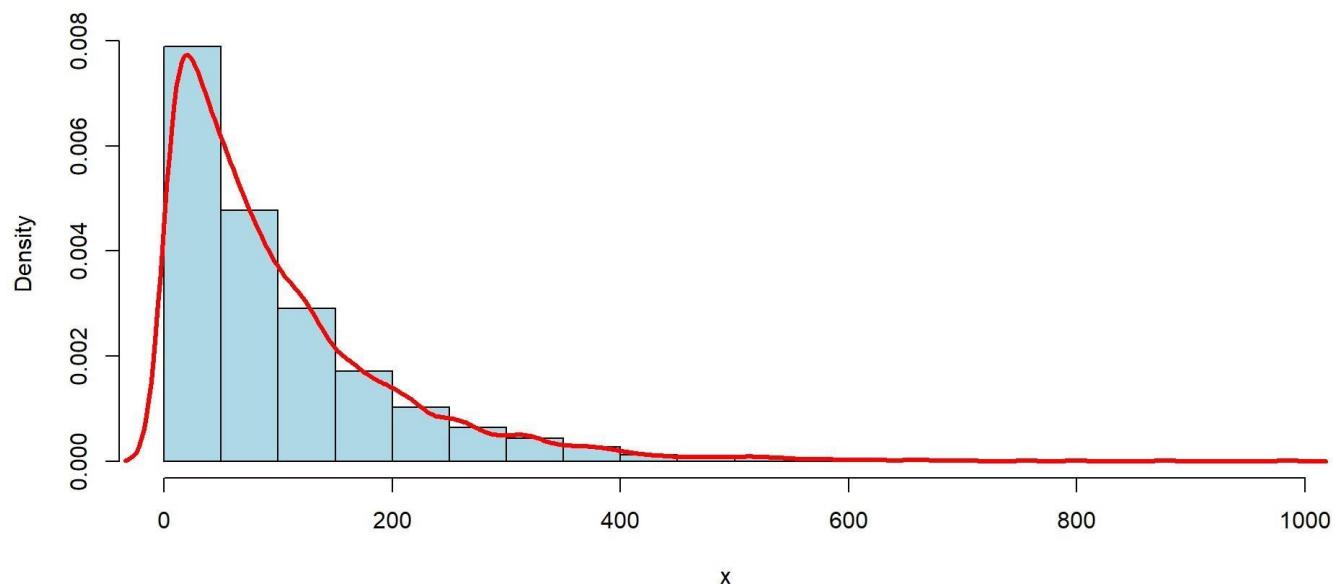
```
N <- 10000 x <-
rgeom(N, .5) hist(x,
  xlim=c(min(x),max(x)), probability=T, nclass=max(x)-min(x)+1,
  col='lightblue',
  main='Geometric distribution, p=.5') lines(density(x,bw=1),
  col='red', lwd=3)
```





```
N <- 10000
x <- rgeom(N, .01)
hist(x,
      xlim=c(min(x),max(x)), probability=T, nclass=20,
      col='lightblue', main='Geometric distribution, p=.01')
lines(density(x), col='red', lwd=3)
```



Geometric distribution, p=.01

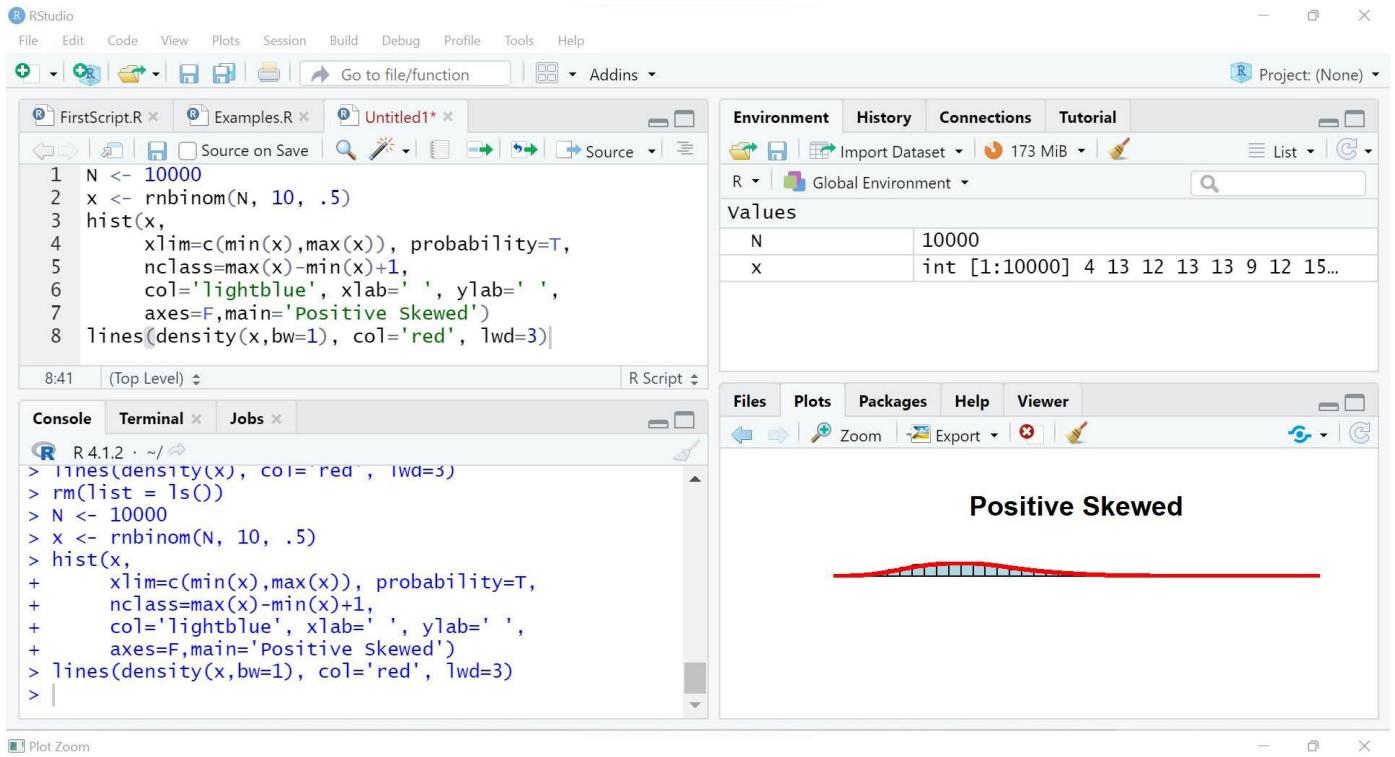
□ Skewer tailed histograms

- For skewed distributions, it is quite common to have one tail of the distribution considerably longer or drawn out relative to the other tail.
- A "skewed right" distribution is one in which the tail is on the right side.
- A "skewed left" distribution is one in which the tail is on the left side.

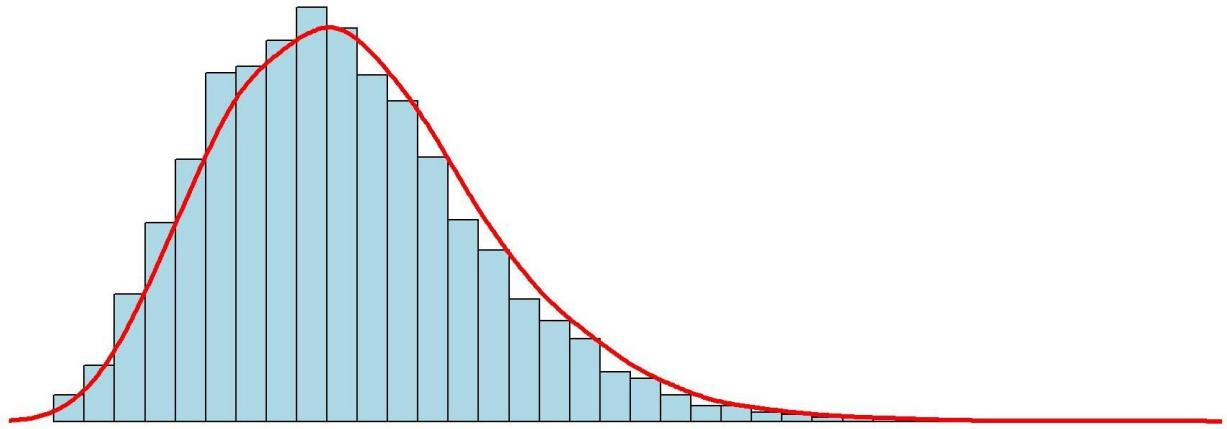
Example Code 1:

```
N <- 10000 x <-
rnbnom(N, 10, .5) hist(x)

xlim=c(min(x),max(x)), probability=T,
nclass=max(x)-min(x)+1,      col='lightblue',
xlab=' ', ylab=' ',      axes=F,main='Positive
Skewed') lines(density(x,bw=1), col='red', lwd=3)
```



Positive Skewed



Example Code 2:

```

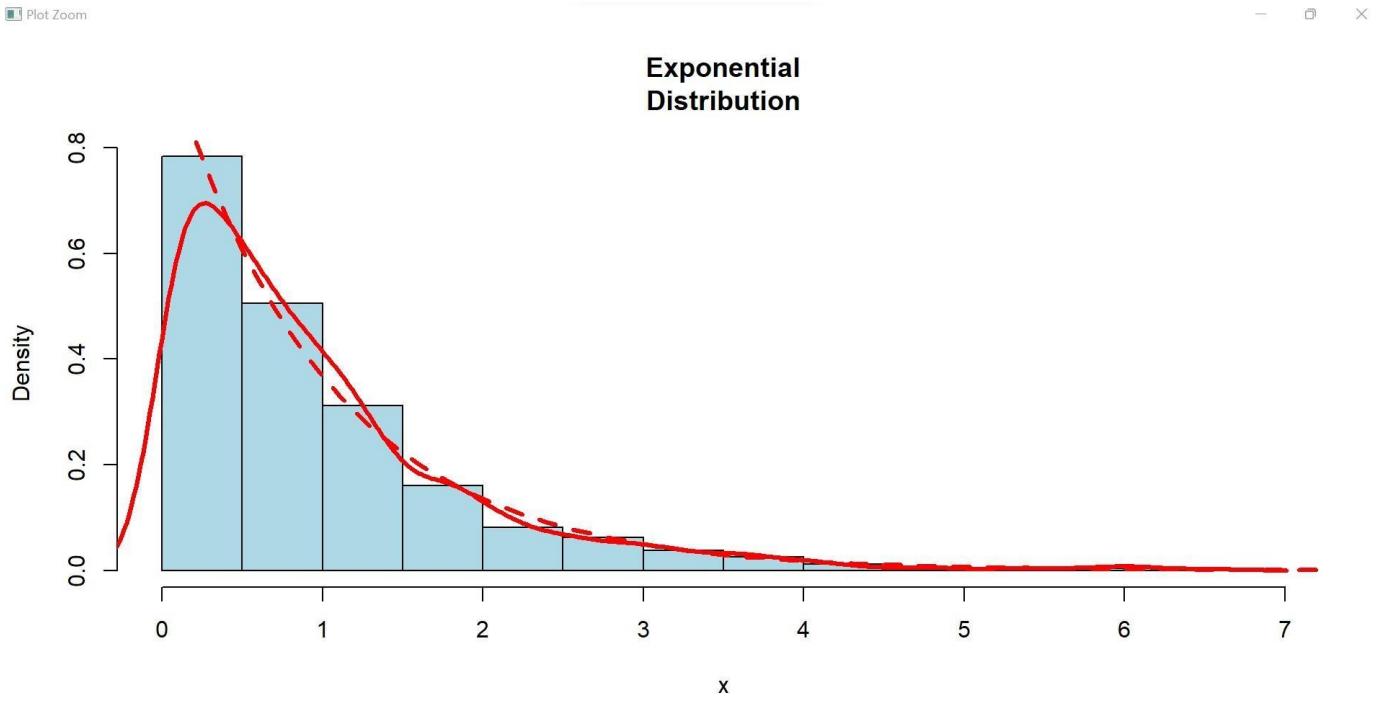
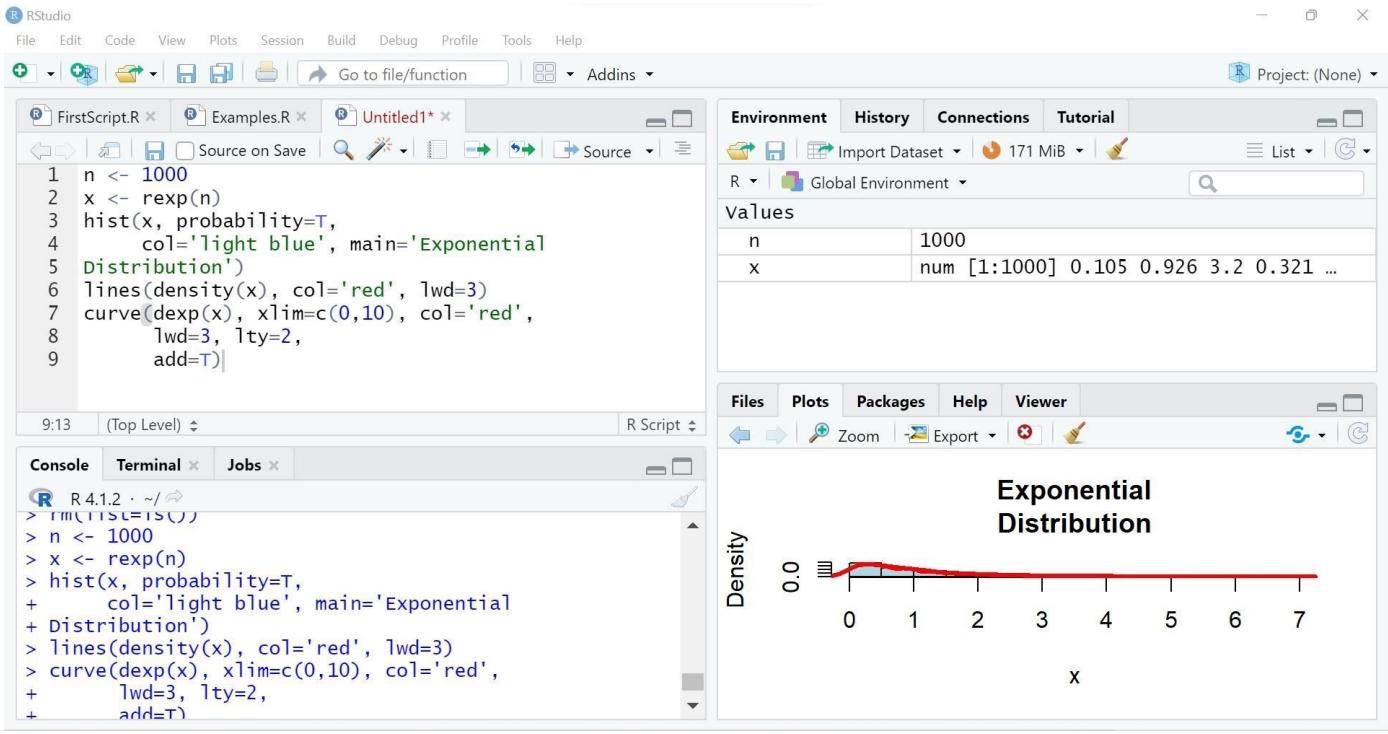
n <- 1000 x <- rexp(n)
hist(x, probability=T,
      col='light blue', main='Exponential Distribution')

```

```

lines(density(x), col='red', lwd=3)
curve(dexp(x), xlim=c(0,10), col='red',
lwd=3, lty=2,      add=T)

```



Q2: Write a R program to plot multiple time series using ggplot2 data visualization package.

Ans:

ggplot2: ggplot2 is an R package used for statistical computing and data representation using data visualization. It follows underlying graphics called Grammar of Graphics which includes

certain rules and independent components which can be used to represent data in various formats.

Program using ggplot2 (step wise)

- We first need to install and load the reshape2 package, if we want to use the functions that are included in the add-on package:
- Now, we can reshape our data from wide to long format using the melt function:
- Now, we need to install and load the ggplot2 package to draw our time series plot using ggplot2:
- Finally, we can apply the ggplot and geom_line commands to draw multiple time series to a plot:

Example Code 1: library("reshape2")

```
library("ggplot2") set.seed(1023172)
```

```
data <- round(data.frame(year = 2001:2025,
```

```
ts1 = 1:25 + rnorm(25),
```

```
ts2 = 30:6 + runif(25, 0, 10), ts3 =
```

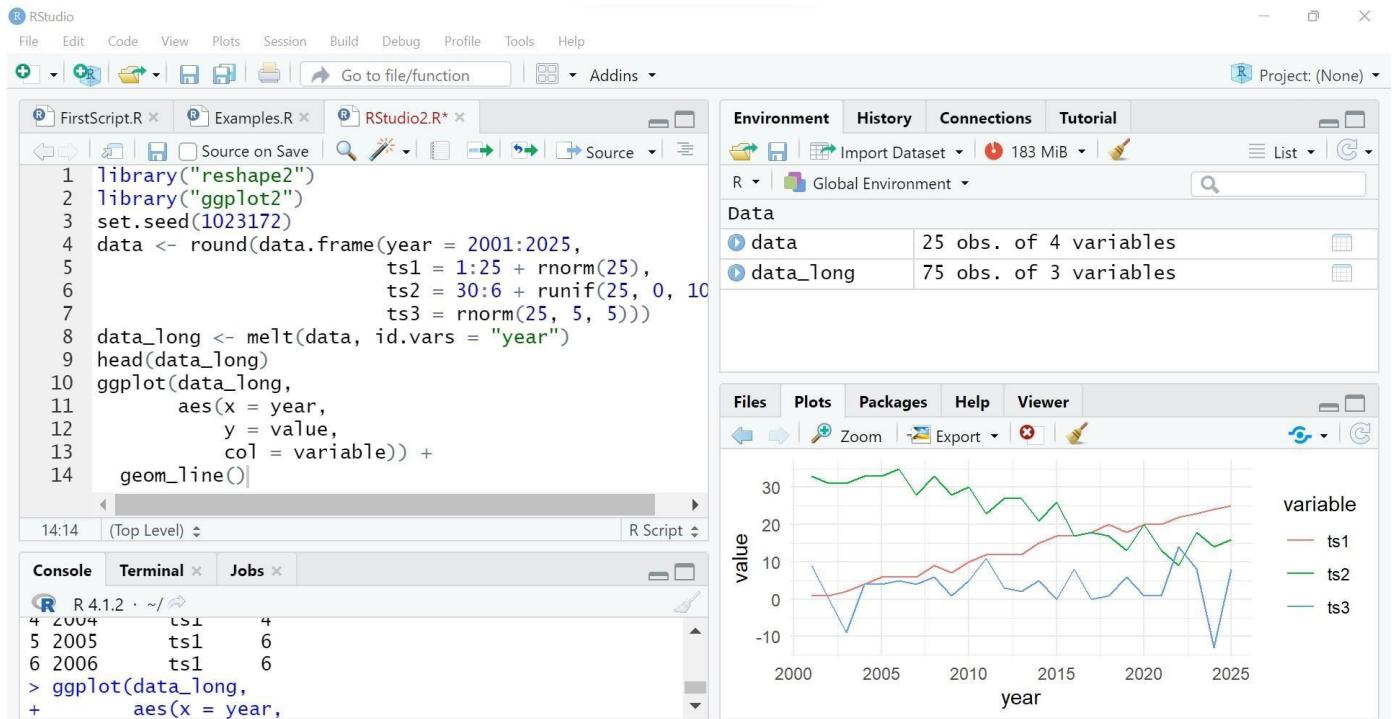
```
rnorm(25, 5, 5))) data_long <- melt(data, id.vars =  
"year") head(data_long)
```

```
ggplot(data_long,
```

```
aes(x = year, y =
```

```
value, col =
```

```
variable)) + geom_line()
```





```
Example code 2: library(ggplot2)
library(reshape2) library(dplyr)
```

```
covid1 =(read.csv(file="EUCOVIDdeaths.csv",header=TRUE)[,-c(2)])
```

```
head(covid1)
```

```
covid_deaths <- melt(covid1,id.vars=c("Country"),value.name="value",
variable.name="Day")
```

```
head(covid_deaths)
```

```
covid_plot <- ggplot(data=covid_deaths, aes(x=Day, y=value, group = Country, colour =
Country)) + geom_line() + labs(y= "Deaths", x = "Day") covid_plot +
ggtitle("Daily Deaths for European countries in
March,2020") +geom_point()
```

```
covid_plot
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

```
es.R RStudio2.R* RNotebook.Rmd timeseries.rmd
14+ ``{r}
15 covid <- read.csv(file="EUCOVIDdeaths.csv",header=TRUE)
16+
5:4 # Title ▾ R Markdown ▾
```

Console Terminal Jobs

```
R 4.1.2 · D/ ↵
2 Belgium Mar16 0
3 Bulgaria Mar16 0
4 Croatia Mar16 0
5 Cyprus Mar16 0
6 Czechia Mar16 0
>
> covid_plot <- ggplot(data=covid_deaths, aes(x=Day, y=value, group = Country,
+                                         colour = Country))+ geom_line() +labs(y= "Deaths", x = "Day")
> covid_plot + ggttitle("Daily Deaths for European countries in March,2020") +geom_point()
>
> covid_plot
>
> |
```

Environment History Connections Tutorial

Import Dataset 283 MiB Global Environment

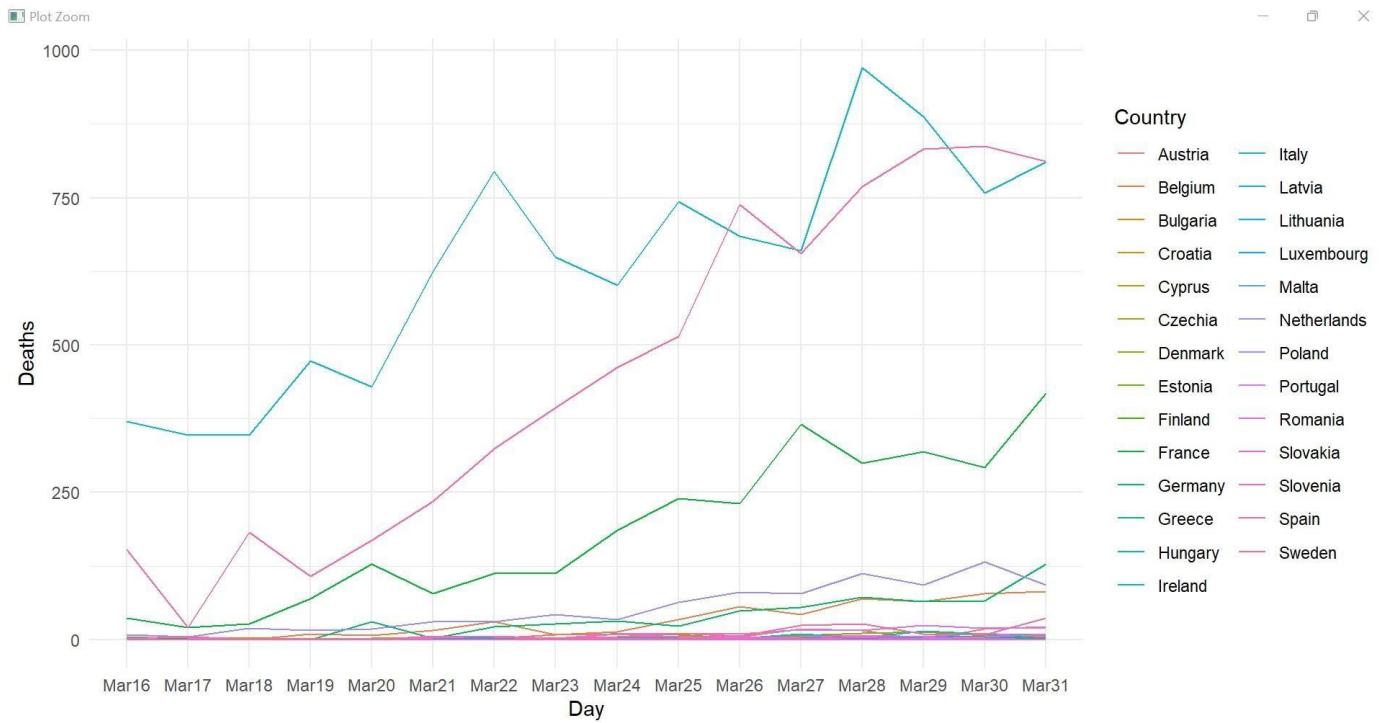
Data

- covid_deaths 432 obs. of 3 variables
- covid_plot List of 9
- covid1 27 obs. of 17 variables

Files Plots Packages Help Viewer

Deaths Day

Croatia Luxembourg
Cyprus Malta
Czechia Netherlands
Denmark Poland
Estonia Portugal
Finland Romania
France Slovakia
Germany Slovenia
Greece Spain
Hungary Ireland



LAB ASSIGNMENT A4:

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

```

1 #Exporting data from it R
2 #Tushar Rath
3 #scholar id : 2012174
4
5 #Importing readr library
6 library(readr)
7
8 #Creating a dataframe
9 df=data.frame(
10
11   "Name"=c ("Naved", "Pranav", "Neha", "Jay", "Arun", "Ram"),
12   "Age"=c(19, 21, 20, 19, 10),
13   "Department"=c ("IT", "B.Sc.", "Agriculture", "Law", "M.Sc", "B. Pharma"),
14   "Salary"=c(500, 500, 700, 800, 900, 1000)
) #Export a data frame sing write_csv()
write.csv(df, file= "My_Export_Data.csv")

```

In the top-right pane, the Environment tab shows the global environment with a data frame named 'df' containing 6 observations and 4 variables: Name (character vector) and Salary (numeric vector). The Data tab shows the same information.

The bottom-right pane is a file browser showing the contents of the current directory, including files like 'RData', 'History', 'ASSIGNMENT 1.docx', and 'My_Export_Data.csv'.

The bottom-left pane is the R Console, which displays the same R code and its execution results.

The screenshot shows a Microsoft Excel spreadsheet titled '1000'. The data is organized into columns A through W. The first row contains column headers: 'Name', 'Age', 'Department', and 'Salary'. Subsequent rows contain data points, with the last row (row 7) being highlighted in green. The 'Salary' column for the last row contains the value '1000'.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		Name	Age	Department	Salary																		
2	1 Naved		19 IT		500																		
3	2 Pranav		21 B.sc		500																		
4	3 Neha		20 Agriculture		700																		
5	4 Jay		20 Law		800																		
6	5 Arun		19 M.sc		900																		
7	6 Ram		10 B.tech		1000																		
8																							
9																							
10																							
11																							
12																							
13																							
14																							
15																							
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							
27																							

At the bottom, the status bar shows 'Sheet1' and 'Accessibility: Good to go'.

```

1 #Exporting data from it R
2 #Tushar Rathi
3 #scholar id : 2021274
4
5 #Importing readr library
6 library(readr)
7
8 #creating a data frame
9 df<-data.frame(
10
11 "Name"=c ("Naved", "Pranav", "Neha", "Jay", "Arun", "Ram"),
12 "Age"= c(19, 21, 20, 20, 19, 10),
13 "Department"=c("IT", "B.Sc.", "Agriculture", "LAW", "M.Sc", "B. Pharma"),
14 "Salary"=c(500, 500, 700, 800, 900, 1000)
15
16 #Export a data frame sing write.csv()
17 write.csv(df, file= "My_Export_Data.csv")
18
19 #Export a data frame sing write.csv()
20 write.csv(df, file= "My_Export_Data.csv")
21
22 #Reading the Data
23 data01 <-read.csv("My_Export_Data.csv")
24
25

```

11:44 (Top Level) R Script

Console Terminal Jobs

```

#creating a dataframe
df<-data.frame(
  "Name"=c ("Naved", "Pranav", "Neha", "Jay", "Arun", "Ram"),
  "Age"= c(19, 21, 20, 20, 19, 10),
  "Department"=c("IT", "B.Sc.", "Agriculture", "LAW", "M.Sc", "B. Pharma"),
  "Salary"=c(500, 500, 700, 800, 900, 1000)
)
#Export a data frame sing write.csv()
write.csv(df, file= "My_Export_Data.csv")

#reading the data
data01 <-read.csv("My_Export_Data.csv")

# display the data
data 1
X Name Age Department Salary
1 Naved 19 500
2 Arun 21 B.Sc. 500
3 Neha 20 Agriculture 700

```

Environment History Connections Tutorial

Data

Name	Value
df	6 obs. of 4 variables
Name	chr [1:6] "Aditya" "Pranav" "Neha" "Nikita" "Vikas"
Salary	num [1:6] 500 500 700 800 900 1000

Files Plots Packages Help Viewer

11:01 PM 3/6/2022

Q2. A) .gz files:

`gzfile(description, open = "", encoding =getOption("encoding"), compression = 6)`

Internally `gzfile()` (see `connections`) is used to read (write) chunks to (from) the gzip file. If the process is interrupted before completed, the partially written output file is automatically removed.

For `gzfile` the description is the path to a file compressed by gzip: it can also open for reading uncompressed files and those compressed by bzip2, xz or lzma.

R Code to read .gz files:

```

>untar("wp2011-survey.tar.gz", list = TRUE)
>untar("wp2011-survey.tar.gz", files = "wp2011-survey/anon-data.csv")
>data <- read.csv("wp2011-survey/anon-data.csv")
>head(data)

```

```
RGu (64-bit) - [R Console]
File Edit View Msc Packages Windows Help
[1] > untar("wp2011-survey.tar.gz", list = TRUE)
[1] "wp2011-survey"
[2] "wp2011-survey/_anon-data.csv"
[3] "wp2011-survey/_questions.txt"
[4] "wp2011-survey/questions.txt"
[5] "wp2011-survey.tar.gz"
> untar("wp2011-survey.tar.gz", files = "wp2011-survey/anon-data.csv")
> X <- read.csv("wp2011-survey/anon-data.csv")
> head(X)
 1
 2 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 3 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 4 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 5 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 6
      job_type
 1
 2     My primary job is as a self-employed developer that uses WordPress.
 3     My primary job is working for a company or organization that uses WordPress.
 4 Work that I do involving WordPress is just a hobby, I don't make money from it.
 5     My primary job is as a self-employed developer that uses WordPress.
 6
 1
 2
 3 Build and/or maintain websites or blogs for other people, companies, or organizations., Build and/or maintain websites or blogs for my own use., Develop or customize themes., Host websit
 4
 5
 6
      c_cms_blog
 1
 2
 3 Mostly as a content management system (CMS).
 4
 5
 6
      c_customize
 1
 2
 3 A lot of work has been done, the front end is unrecognizable, but the Dashboard still looks like the usual WordPress interface.
 4
 5
 6
      c_number c_percent          c_done_with_wp c_living
 1 |
```

B) .bz2 files:

Similarly to read .bz2 files **untar()** function is also used. The choice of compression whether ".gz" or ".bz2" is passed as a parameter in both the functions. In **untar()** the parameter is called compressed and is deprecated in favour of auto detection of the compression technique.

R Code to read .bz2 files:

```
>untar("wp2011-survey.tar.bz2", list = TRUE)
>untar("wp2011-survey.tar.bz2", files = "archive/wp2011-survey/anon-c
>X <- read.csv("archive/wp2011-survey/anon-data.csv")
>head(X)
```

```
[3] "archive/wp2011-survey/questions.txt"  "archive/wp2011-survey/anon-data.csv"
[5] "archive/wp2011-survey/.questions.txt"  "archive/wp2011-survey/.anon-data.csv"
> unter("wp2011-survey.tar.bz2", files = "archive/wp2011-survey/anon-data.csv")
> A <- read.csv("archive/wp2011-survey/anon-data.csv")
> head(A)
 1
 2 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 3 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 4 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 5 A. I'm a developer, or I work for a company that develops websites; I use WordPress to build websites and/or blogs for others. (This might include theme development, writing plugins, etc)
 6 B. I own, run, or contribute to a blog or website that is built
 7
 8 My primary job is as a self-employed developer that uses WordPress.
 9
 10 My primary job is working for a company or organization that uses WordPress.
 11 Work that I do involving WordPress is just a hobby, I don't make money from it.
 12
 13 My primary job is as a self-employed developer that uses WordPress.
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23 Build and/or maintain websites or blogs for other people, companies, or organizations., Build and/or maintain websites or blogs for my own use., Develop or customize themes., Host websit
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 580
 581
 582
 583
 584
 585
 586
 587
 588
 589
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 679
 680
 681
 682
 683
 684
 685
 686
 687
 688
 689
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 779
 780
 781
 782
 783
 784
 785
 786
 787
 788
 789
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 879
 880
 881
 882
 883
 884
 885
 886
 887
 888
 889
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1279
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1289
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298
 1299
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1309
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1319
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1379
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1499
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1519
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1529
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1539
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1549
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1569
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1579
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1589
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1599
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1619
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1679
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1789
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1959
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1969
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1979
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1989
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1996
 1997
 1998
 1999
 1999
 2000
 2001
 2002
 2003
 2004
 2005
 2006
 2007
 2008
 2009
 2009
 2010
 2011
 2012
 2013
 2014
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023
 2024
 2025
 2026
 2027
 2028
 2029
 2029
 2030
 2031
 2032
 2033
 2034
 2035
 2036
 2037
 2038
 2039
 2039
 2040
 2
```

C) URLs:

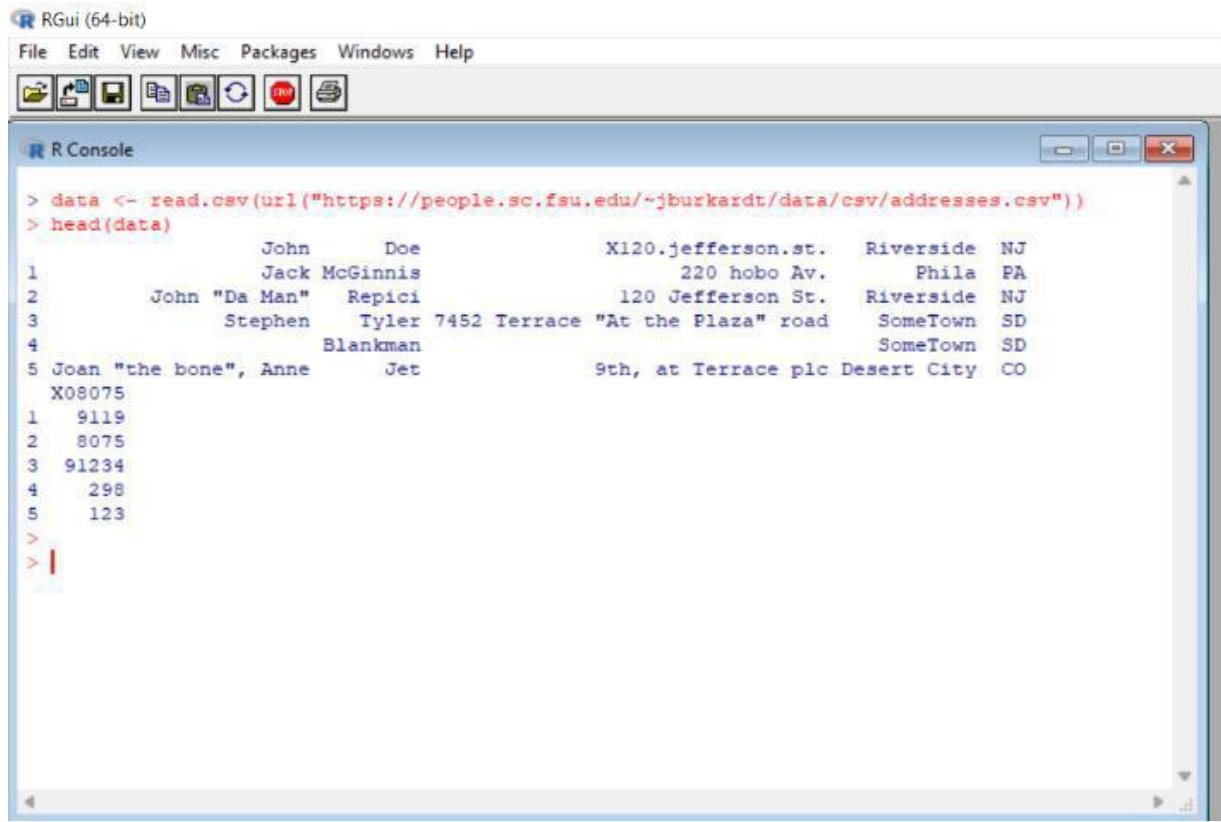
For URL the description is a complete URL including scheme (such as 'http://', 'https://', 'ftp://' or 'file://'). Method "internal" is that available since connections were introduced but now mainly defunct.

Function used:

```
url(description, open = "", blocking = TRUE, encoding =getOption("encoding"),
method =getOption("url.method", "default"), headers = NULL)
```

R Code to read URLs:

```
>data <-  
read.csv(url("https://people.sc.fsu.edu/~jburkardt/data/csv/addresses.csv"))  
>head(data)
```



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> data <- read.csv(url("https://people.sc.fsu.edu/~jburkardt/data/csv/addresses.csv"))
> head(data)
      John Doe           X120.jefferson.st. Riverside NJ
1      Jack McGinnis          220 hobo Av.     Phila PA
2      John "Da Man" Repici       120 Jefferson St. Riverside NJ
3      Stephen Tyler 7452 Terrace "At the Plaza" road SomeTown SD
4      Blankman                9th, at Terrace plc Desert City CO
5 Joan "the bone", Anne Jet
X08075
1 9119
2 8075
3 91234
4 298
5 123
>
>
```

D) MS Excel files:

To import MS Excel files one can use:

```
>my_data <- read.table(file = "clipboard",
+sep = "\t", header = TRUE)
```

OR

```
>library("readxl")
># For xls files
>my_data <- read_excel("my_file.xls")
># For xlsx files
>my_data <- read_excel("my_file.xlsx")
```

For the 2nd procedure we need to install the readxl package using
`install.packages("readxl")`.

R Code for reading MS Excel files:

```
library("readxl")
my_data <- read_excel("Q2.xlsx")
```

```
RGui (64-bit)
File Edit Packages Windows Help
R Console

> library("readxl")
> my_data <- read_excel("Q2.xlsx")
> head(my_data)
# A tibble: 6 x 3
  job.title     city   average.income
  <chr>        <chr>      <dbl>
1 Product Manager San Francisco    85000
2 Software Developer San Francisco 112000
3 Accountant     Austin          60000
4 Accountant     San Francisco  68000
5 Product Manager New York City  78000
6 Software Developer Austin         82000
>
```

E) Binary files:

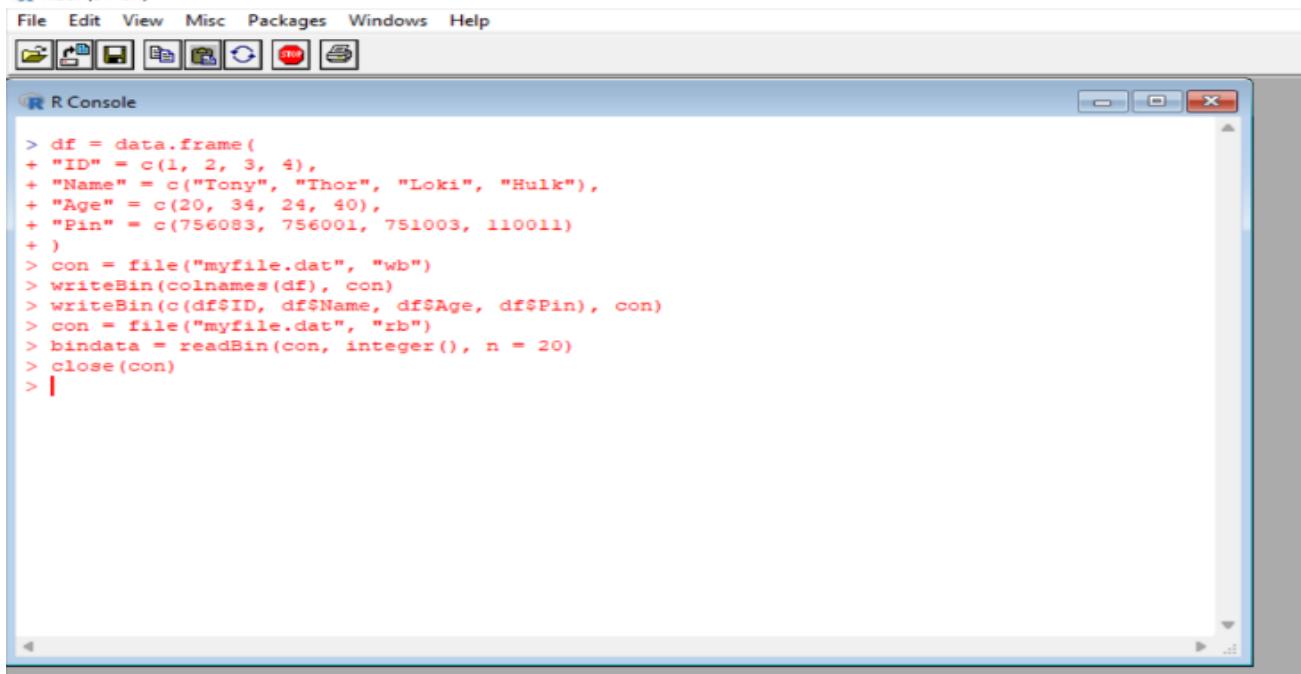
The both creating and writing to a binary file can be performed by a single function **writeBin()** by opening the file in "wb" mode where w indicates write and b indicates binary mode. The Syntax is writeBin(object, con). Con is a connection object or a character string naming a file or a raw vector etc.

R Code for Writing Binary files:

```
>df = data.frame(
+ "ID" = c(1, 2, 3, 4),
+ "Name" = c("Tony", "Thor", "Loki", "Hulk"),
+ "Age" = c(20, 34, 24, 40),
+ "Pin" = c(756083, 756001, 751003, 110011)
>)
>
># Creating a connection object to write the binary file using mode "wb"
>con = file("myfile.dat", "wb")
>
># Write the column names of the data frame to the connection object
>writeBin(colnames(df), con)
>
># Write the records in each of the columns to the file
>writeBin(c(df$ID, df$name, df$Age, df$Pin), con)
```

```

>
># Close the connection object
>close(con)

RGui (64-bit)
File Edit View Misc Packages Windows Help


```

F) ASCII files:

The function used here to write an ASCII file is `write.csv()` and `write.table()`. To save the object in ASCII format we use `saveRDS(data, "data.rds", ascii = TRUE)`.

R Code to write ASCII files:

```

>data <- read.table(header=TRUE, text='
+ subject sex size
+ 1 M 7
+ 2 F NA
+ 3 F 9
+ 4 M 11
+') 
># Write to a file, suppress row names
>write.csv(data, "data.csv", row.names=FALSE)
># Same, except that instead of "NA", output blank cells
>write.csv(data, "data.csv", row.names=FALSE, na="")
># Use tabs, suppress row names and column names
>write.table(data, "data.csv", sep="\t", row.names=FALSE, col.names=FALSE)
>saveRDS(data, "data.rds", ascii = TRUE)
># To load the data again:
>data <- readRDS("data.rds")
>head(data)

```

```

> # A sample data frame
> data <- read.table(header=TRUE, text='
+   subject age height
+   1      21     7
+   2      22    NA
+   3      54     9
+   4      33    11
+ ')
>
>
> # Write to a file, suppress row names
> write.csv(data, "data.csv", row.names=FALSE)
>
> # Same, except that instead of "NA", output blank cells
> write.csv(data, "data.csv", row.names=FALSE, na="")
>
> # Use tabs, suppress row names and column names
> write.table(data, "data.csv", sep="\t", row.names=FALSE, col.names=FALSE)
>
> # Or, using ASCII format
> saveRDS(data, "data.rds", ascii=TRUE)
> data <- readRDS("data.rds")
> head(data)
  subject age height
1       1   21     7
2       2   22    NA
3       3   54     9
4       4   33    11
> |

```

LAB ASSIGNMENT A5:

1. What is KS Test?

The Kolmogorov-Smirnov Test is a type of nonparametric test of the equality of discontinuous and continuous of a 1D probability distribution that is used to compare the sample with the reference probability test (known as one-sample K-S Test) or among two samples (known as two-sample K-S test).

It quantifies a distance between the cumulative distribution function of the given reference distribution and the empirical distributions of given two samples, or between the empirical distribution of given two samples.

In a one-sample K-S test, the distribution that is considered under a null hypothesis can be purely discrete or continuous or mixed. In the two-sample K-S test, the distribution considered under the null hypothesis is generally continuous distribution but it is unrestricted otherwise.

Kolmogorov-Smirnov Test Formula

The formula for the Kolmogorov-Smirnov test can be given as:

$D_n = \sup_x |F_n(x) - F(x)|$ where, \sup_x : the supremum of the set of distances

$F(x)$: the empirical distribution function for n id observations X_i .

The empirical distribution function is a distribution function that is associated with the empirical measures of the chosen sample. Being a step function, this cumulative distribution jumps up by a $1/n$ step at each and every n data points.

2. What is the Necessity to perform KS Test?

The Kolmogorov-Smirnov test is a nonparametric goodness-of-fit test and is used to determine whether two distributions differ, or whether an underlying probability distribution differs from a hypothesized distribution. It is used when we have two samples coming from two populations that can be different.

3. Steps to perform KS Test in R

The K-S test can be performed using the `ks.test()` function in R.

Syntax: `ks.test(x, y, ..., alternative = c("two.sided", "less", "greater"), exact = NULL, tol = 1e-8, simulate.p.value = FALSE, B = 2000)`
 Parameters: `x`: numeric vector of data values `y`: numeric vector of data values or a character string which is used to name a

cumulative distribution function. ...: the parameters which are defined by the y value alternative: used to indicate the alternate hypothesis.

exact: usually NULL or it indicates a logic that an exact pvalue should be computed.

tol: an upper bound used for rounding off errors in the data values.

simulate.p.value: a logic that checks whether to use Monte Carlo method to compute the p-value.

B: an integer value that indicates the number of replicates to be created while using the Monte Carlo method. Step 1: At first install the required packages. For performing the K-S test we need to install the "dgof" package using the install.packages() function from the R console.

```
install.packages("dgof")
```

Step 2: After a successful installation of the package, load the required package in our R Script. for that purpose, use the library() function as follows:

```
# loading the required package library("dgof")
```

Step 3: Use the rnorm() function and the runif() function to generate two samples say x and y. The rnorm() function is used to generate random variates while the runif() function is used to generate random deviates.

```
# loading the required package library(dgof)
```

```
# generating random variate sample 1 x
```

```
<- rnorm(50)
```

```
# generating random deviates sample 2 y <- runif(30)
```

Step 4: Now perform the K-S test on these two samples. For that purpose, use the ks.test() of the dgof package. # loading the required package

```
library(dgof)
```

```
# generating random variate # sample 1 x
```

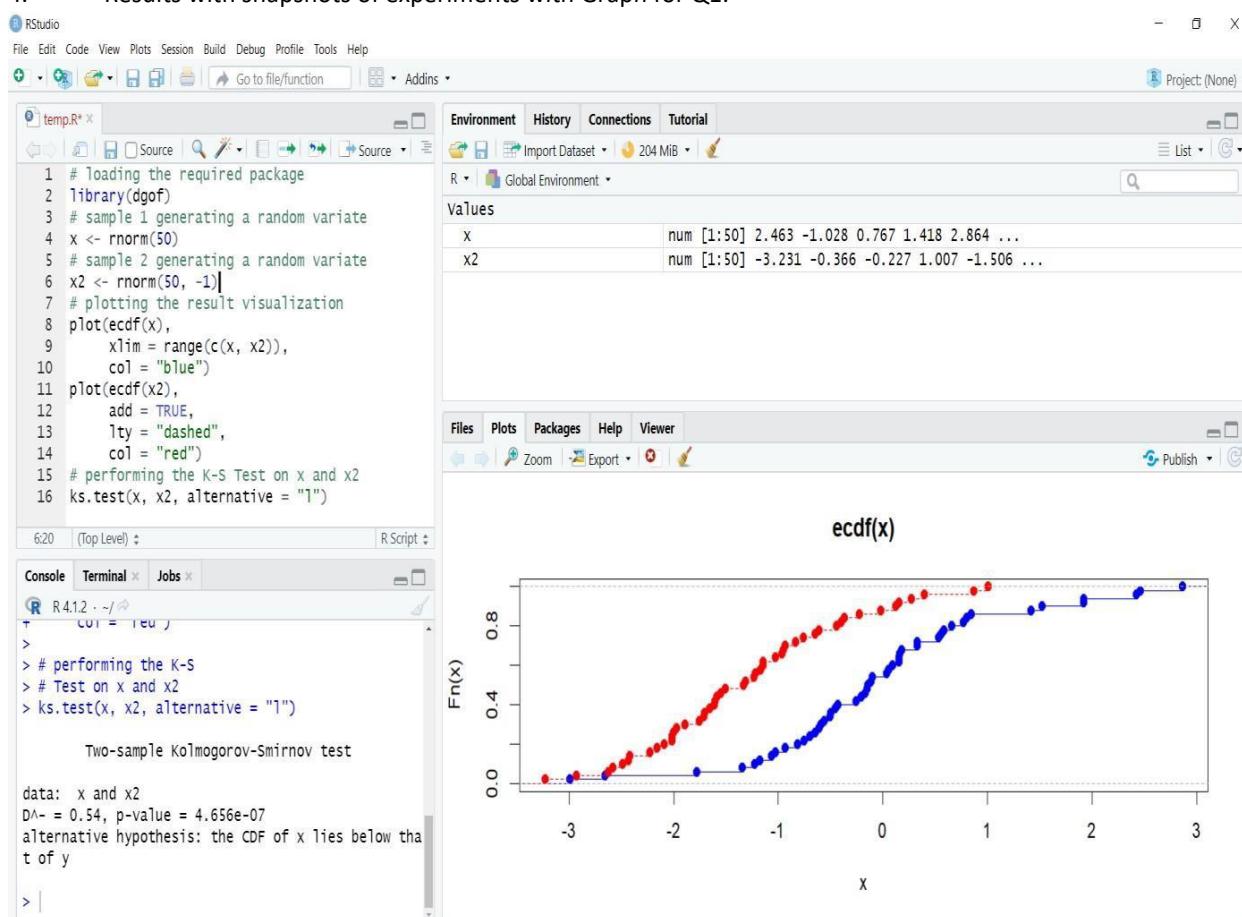
```
<- rnorm(50)
```

```
# generating random deviates # sample 2 y
```

```
<- runif(30)
```

```
# performing the K-S Test # Do x and y come from # the same distribution? ks.test(x, y)
```

4. Results with snapshots of experiments with Graph for Q1.



a) The Stat students of a reputed college have the following test scores: 58, 64, 93, 71, 67, 85, 71, 93, 82, 81, 75, 78, 86, 90, and 87.

Determine and interpret the Quartiles of these scores.

Answer:

1. To determine Quartiles of given scores use the quantile () in R console.
2. For interpretation after calculating Quartiles based on results we analyze our results.
3. For Eg. If Q=0% and corresponding value is 60 means no scores fall below the score 60. If Q=50% means, from the given scores half of them are located above and half will be located below the score 50. Likewise you have to interpret the results.

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and a Project dropdown set to (None). The left pane contains a code editor with a script named 'temp.R' containing the following R code:

```
1 x<- c(58, 64, 93, 71, 67, 85, 71, 93, 82, 81, 75, 78, 86, 90, 87)
2 res<-quantile(x,probs=c(0,.25,.5,.75,1))
3 res
```

The right pane shows the Environment view with the following data:

res	Named num [1:5] 58 71 81 86.5 93
x	num [1:15] 58 64 93 71 67 85 71 93 82 81 ...

The bottom pane is the Console, showing the R session output:

```
R 4.1.2 - ~/d/
> x<- c(58, 64, 93, 71, 67, 85, 71, 93, 82, 81, 75, 78, 86, 90, 87)
> res<-quantile(x,probs=c(0,.25,.5,.75,1))
> res
 0% 25% 50% 75% 100%
58.0 71.0 81.0 86.5 93.0
>
```

Ananlysis:

Therefore Q1=0% implies that, none of the scores fall below or equal to 58.0.

Q2=25%, implies that one-fourth of the data are below or equal to 71.0

Q3=50% is the median, and thus half of the scores are below or equal to 81.0, while the other half, are above 81.0

Q4=75%, implies that three-fourth of the data are below or equal to 86.5, while the remaining one-fourth are above 86.5. And the minimum and maximum values are 58.0 and 93.0, respectively.

b) The surveyed weights (in kilogram) of the students in Stat dept. were the following: 50, 65, 72, 62, 77, 84, 57, 74, 66, 68, 75, 58, 52, 69, and 87.

Compute and interpret the Deciles of these weights.

Answer:

1. Load the weights of the students in with a variable in R.
2. Calculate deciles with quantile(weights, prob = seq(0, 1, length =), type =)

From the above function

Weights - weights of the students.

Prob - position of the data point.

Seq - used for prob's value that is from 0 to 1 of length length - number of weights in addition with 0, because the value 0 is the minimum of the data points.

The screenshot shows the RStudio interface. In the top-left pane, a script named 'temp.R' contains the following code:

```

1 weights <- c(50, 65, 72, 62, 77, 84, 57, 74, 66, 68, 75, 58, 52, 69, 87)
2 quantile(weights, prob = seq(0, 1, length = 11), type = 5)
3
4

```

In the bottom-left pane, the 'Console' tab shows the output of the command:

```

> weights <- c(50, 65, 72, 62, 77, 84, 57, 74, 66, 68, 75, 58, 52, 69, 87)
> quantile(weights, prob = seq(0, 1, length = 11), type = 5)
  0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
50.0 52.0 57.5 62.0 65.5 68.0 70.5 74.0 76.0 84.0 87.0
>

```

The top-right pane shows the 'Environment' tab with a table of values:

	weights	num [1:15]
50	50	50
65	65	65
72	72	72
62	62	62
77	77	77
84	84	84
57	57	57
74	74	74
66	66	66
68	68	68
75	75	75
58	58	58
52	52	52
69	69	69
87	87	87

Analysis:

The first decile is D1=10%, implies that one-tenth of the weights fall below or equal to 52.0, and the remaining ninetenth fall above 52.0. The D5=50% is the median, thus half of the students' weights weigh below or equal to 68.0, while the other half fall above it. And so on.

c) Compute the power of a study to show a difference between group 1 (n=28) in which the event probability is 30% and group 2 (n=28) in which the event probability is 55%.

Answer:

The screenshot shows the RStudio interface. In the top-left pane, a script named 'temp.R' contains the following code:

```

1 power.prop.test(n=28,p1=0.3,p2=0.55)

```

In the bottom-left pane, the 'Console' tab shows the output of the command:

```

> power.prop.test(n=28,p1=0.3,p2=0.55)
Two-sample comparison of proportions power calculation

  n = 28
  p1 = 0.3
  p2 = 0.55
  sig.level = 0.05
  power = 0.4720963
  alternative = two.sided

NOTE: n is number in *each* group
>

```

The top-right pane shows the 'Environment' tab with the message "Environment is empty".

Conclusion:

The power of a study which includes 28 subjects in each of two experimental groups to see a difference between event probabilities is 48% under the assumption that the event probabilities are 30% in group 1 and 55% in group

2.

d) Compute the sample size of a study to show a difference between group 1 in which the event probability is 25% and group 2 in which the event probability is 45% with a power of 65%.

Answer:

The screenshot shows the RStudio interface. In the top-left pane, there is a script editor window titled "temp.R" containing the following R code:

```
1 power.prop.test(power=0.65,p1=0.25,p2=0.45)
2
```

In the bottom-left pane, the "Console" tab is active, showing the command and its output:

```
R 4.1.2 - ~/ ◊
> power.prop.test(power=0.65,p1=0.25,p2=0.45)

Two-sample comparison of proportions power calculation

n = 62.1105
p1 = 0.25
p2 = 0.45
sig.level = 0.05
power = 0.65
alternative = two.sided

NOTE: n is number in "each" group
>
```

The right side of the interface shows the "Environment" and "Files" panes, both of which are currently empty.

Conclusion: In order to achieve a power of 65% under the assumed event probabilities the study should include at least 63 subjects in each of the experimental groups.

LAB ASSIGNMENT-A6

1. Write a R Program based on linear regression model on given data set X=height and Y=weight, where predicted weight of a person having height is 155.

Dataset: -

Height	151	174	130	140	150
Weight	63	82	48	58	60

Answer: R-Code snippet:

```
height<-
c(151,174,130,140,150)
weight<-c(63,82,48,58,60)
#Using lm to get the relation between the height and weight
relation<-lm(weight~height)
print(relation)
#Predicting the weight of person with height
155 a<-data.frame(height=155) result<-
predict(relation,a) print(result)
```

The screenshot shows the RStudio interface with the following components:

- Code Editor (temp.R):** Contains the R code provided in the answer section.
- Environment pane:** Shows the global environment with objects `a` (1 obs. of 1 variable), `relation` (List of 12), and `values` (height, result, weight).
- Console pane:** Displays the R session output, including the execution of the code and the resulting prediction for a height of 155.

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function | Addins |
Project: (None) ▾

temp.R x
1 height<-c(151,174,130,140,150)
2 weight<-c(63,82,48,58,60)
3 #Using lm to get the relation between the height and weight
4 relation<-lm(weight~height)
5 print(relation)
6 #Predicting the weight of person with height 155
7 a<-data.frame(height=155)
8 result<-predict(relation,a)
9 print(result)
10

10:1 (Top Level) ▾
R Script ▾

Console Terminal Jobs
R 4.1.2 : -/-
Call:
lm(formula = weight ~ height)

Coefficients:
(Intercept)      height
-49.2720        0.7481

> #Predicting the weight of person with height 155
> a<-data.frame(height=155)
> result<-predict(relation,a)
> print(result)
 1
66.68881
>
```

Hence weight of the person with height 155 is **66.68881**.

2. Answer all by examples and with R language specified functions/library if required.

a. Dependent and independent variable:

⑨ The “dependent variable” represents the output or effect, or is tested to see if it is the effect. A dependent variable is the variable that changes as a result of the independent variable manipulation. It’s the outcome you’re interested in measuring, and it “depends” on your independent variable.

⑨ The “independent variables” represent the inputs or causes, or are tested to see if they are the cause. An independent variable is the variable you manipulate or vary in an experimental study to explore its effects. It’s called “independent” because it’s not influenced by any other variables in the study.

Example:

Let us consider $Y=3X+5$, here the value of X is independent of any other variable, but the value of Y is dependent on the value of X , therefore X is independent variable and Y is dependent variable.

b. Dummy variables:

⑨ Dummy variable in R programming is a type of variable that represents a characteristic of an experiment. A dummy variable is either 1 or 0 and 1 can be represented as either True or False and 0 can be represented as False or True depending upon the user. This variable is used to categorize the characteristic of an observation.

⑨ We can create dummy variables in R using 2 methods, `ifelse()` function or by using `dummy_cols()` function.

⑨ Using `ifelse()` function: `ifelse()` function performs a test and based on the result of the test return true value or false value as provided in the parameters of the function. Using this function, dummy variable can be created accordingly.

Syntax:

`ifelse(test, yes, no)`

Parameters:

`test`: represents test condition. `yes`: represents the value which will be executed if test condition satisfies. `no`: represents the value which will be executed if test condition does not satisfies.

⑨ R-Code Snippet:

```
# Using PlantGrowth dataset
pg <- PlantGrowth
# Print
cat("Original dataset:\n")
head(pg, 20)
# Create dummy variable
pg$group_ctrl <- ifelse(pg$group == "ctrl", 1, 0)
# Print cat("After creating dummy variable:\n")
head(pg, 20)
```

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 Open, Save, Run, Source, and Addins.
- Code Editor:** A script named "temp.R" is open, containing R code to load the PlantGrowth dataset, print its head, create a dummy variable "group_ctrl", and print the head again.
- Console:** Shows the execution of the R code, displaying the original dataset's head and the modified dataset's head after creating the dummy variable.
- Environment Tab:** Shows the "pg" object in the Global Environment, which contains 30 observations and 3 variables.
- Data View:** Shows the "pg" dataset with columns "weight", "group", and "group_ctrl".

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 Open, Save, Run, Source, and Addins.
- Code Editor:** A script named "temp.R" is open, containing R code to load the PlantGrowth dataset, print its head, create a dummy variable "group_ctrl", and print the head again.
- Console:** Shows the execution of the R code, displaying the original dataset's head and the modified dataset's head after creating the dummy variable.
- Environment Tab:** Shows the "pg" object in the Global Environment, which contains 30 observations and 3 variables.
- Data View:** Shows the "pg" dataset with columns "weight", "group", and "group_ctrl".

The screenshot shows the RStudio interface. In the top-left, the script editor window titled 'temp.R* x' contains the following R code:

```

1 # Using PlantGrowth dataset
2 pg <- PlantGrowth
3 # Print
4 cat("Original dataset:\n")
5 head(pg, 20)
6 # Create dummy variable
7 pg$group_ctrl <- ifelse(pg$group == "ctrl", 1, 0)
8 # Print
9 cat("After creating dummy variable:\n")
10 head(pg, 20)

```

In the top-right, the 'Data' tab of the 'Environment' panel shows a dataset named 'pg' with 30 observations and 3 variables.

In the bottom-left, the 'Console' tab displays the output of the 'head' command:

```

R 4.1.2 - ~/r/
1 5.58 ctrl 1
2 5.18 ctrl 1
3 6.11 ctrl 1
4 4.50 ctrl 1
5 4.61 ctrl 1
6 5.17 ctrl 1
7 4.53 ctrl 1
8 5.33 ctrl 1
9 5.14 ctrl 1
10 4.81 trtl 0
11 4.17 trtl 0
12 4.41 trtl 0
13 3.59 trtl 0
14 5.87 trtl 0
15 3.83 trtl 0
16 6.03 trtl 0
17 4.89 trtl 0
18 4.32 trtl 0
19 4.69 trtl 0
20
> |

```

⑨Using `dummy_cols()` function

`dummy_cols()` function is present in fast Dummies package. It creates dummy variables on the basis of parameters provided in the function. If columns are not selected in the function call for which dummy variable has to be created, then dummy variables are created for all characters and factors column in the dataframe.

Syntax: `dummy_cols(.data, select_columns`

`= NULL)` Parameters:

`.data`: represents object for which dummy columns has to be created.

`select_columns`: represents columns for which dummy variables has to be created.

Example :

```
# Create a dataframe df <- data.frame(gender = c("m", "f",
"m"), age = c(19, 20, 20),
city = c("Delhi", "Mumbai", "Delhi"))
```

```

# Create dummy variables
# select_columns = NULL uses all
# character and factor columns
# to create dummy variable df
<- dummy_cols(df)
# Print print(df)

```

```

install.packages("fastDummies")
library(fastDummies)
df <- data.frame(gender = c("m", "f", "m"),
                 age = c(19, 20, 20),
                 city = c("Delhi", "Mumbai",
                          "Delhi"))
# Create dummy variables
# select_columns = NULL uses all
# character and factor columns
# to create dummy variable df
df <- dummy_cols(df)
# Print
print(df)

```

```

R 4.1.2 · ~/temp.R
> # Create dummy variables
> # select_columns = NULL uses all
> # character and factor columns
> # to create dummy variable
> df <- dummy_cols(df)
>
> # Print
> print(df)
   gender age city gender_f gender_m city_Delhi city_Mumbai
1      m  19 Delhi        0        1         1          0
2      f  20 Mumbai       1        0         0         1
3      m  20 Delhi       0        1         1          0

```

c. Mean value with an example

❾ It is calculated by taking the sum of the values and dividing with the number of values in a data series. ❾ The function **mean()** is used to calculate this in R.

Syntax:

The basic syntax for calculating mean in R is –

mean(x, trim = 0, na.rm = FALSE, ...) Following is the

description of the parameters used :

❾ **x** is the input vector.

⑨ **trim** is used to drop some observations from both end of the sorted vector.

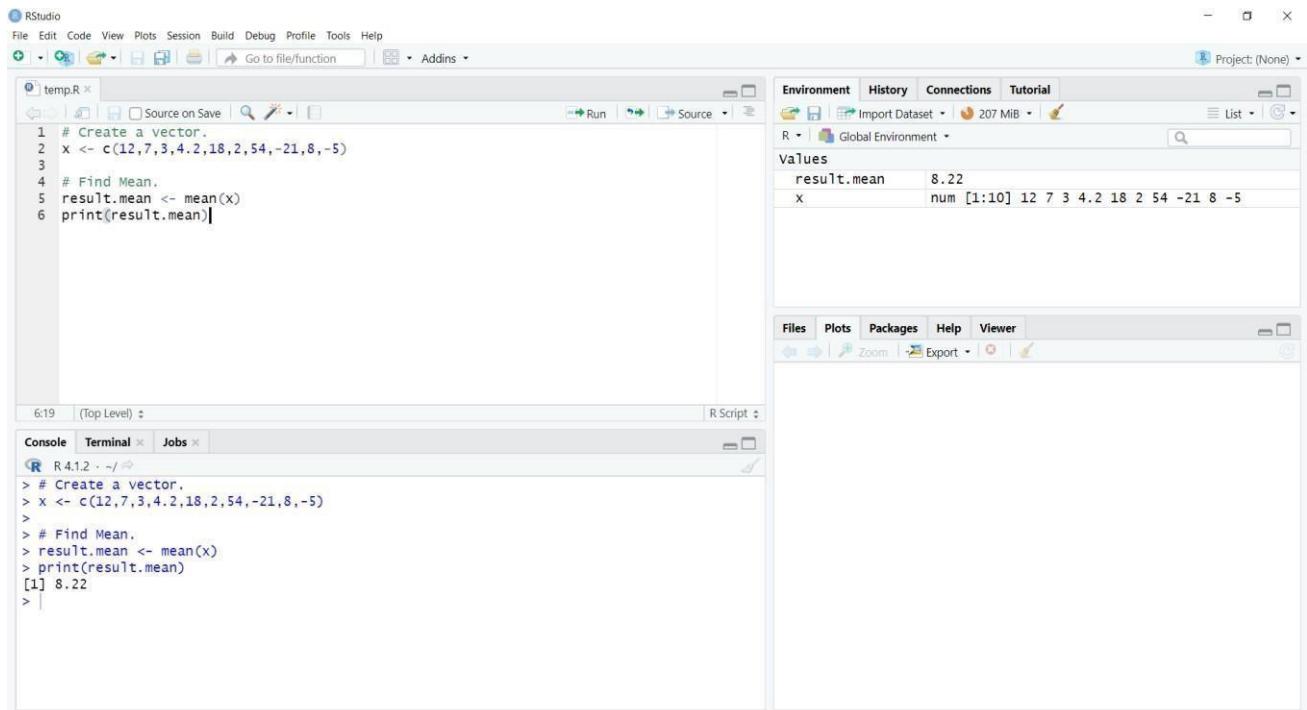
⑨ **na.rm** is used to remove the missing values from the input vector.

Example:

```
# Create a vector. x <-  
c(12,7,3,4.2,18,2,54,-21,8,-5) #
```

Find Mean.

```
result.mean <- mean(x)  
print(result.mean)
```



Applying Trim Option:

When trim parameter is supplied, the values in the vector get sorted and then the required numbers of observations are dropped from calculating the mean.

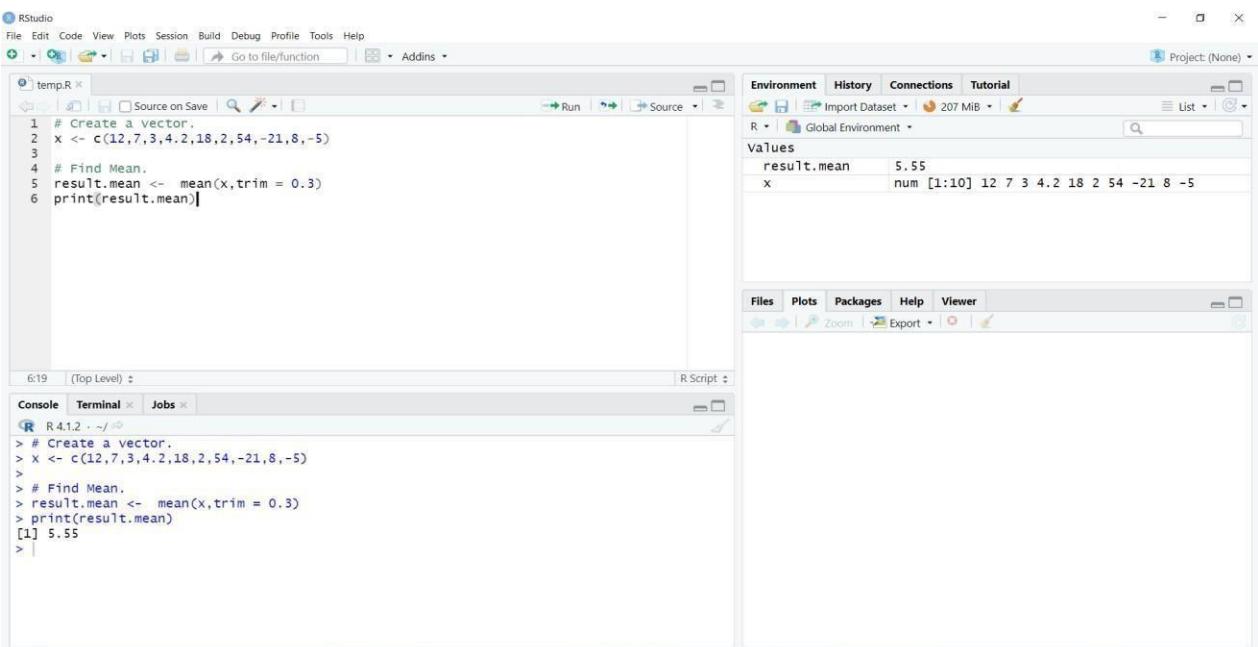
When $\text{trim} = 0.3$, 3 values from each end will be dropped from the calculations to find mean.

In this case the sorted vector is $(-21, -5, 2, 3, 4.2, 7, 8, 12, 18, 54)$ and the values removed from the vector for calculating mean are $(-21, -5, 2)$ from left and $(12, 18, 54)$ from right.

Example:

```
# Create a vector. x <-
c(12,7,3,4.2,18,2,54,-21,8,-5) #
```

```
Find Mean. result.mean <-
mean(x,trim = 0.3)
print(result.mean)
```



The screenshot shows the RStudio interface with the following components visible:

- Code Editor:** Shows the R script file "temp.R" with the following code:

```
1 # Create a vector.
2 x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
3
4 # Find Mean.
5 result.mean <- mean(x,trim = 0.3)
6 print(result.mean)
```
- Console:** Shows the R session output:

```
R 4.1.2 - ~/R
> # Create a vector.
> x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
>
> # Find Mean.
> result.mean <- mean(x,trim = 0.3)
> print(result.mean)
[1] 5.55
> |
```
- Environment View:** Shows the global environment with the variable "result.mean" set to 5.55 and the vector "x" containing the values [1:10] 12 7 3 4.2 18 2 54 -21 8 -5.

Applying NA Option:

If there are missing values, then the mean function returns NA.

To drop the missing values from the calculation use $\text{na.rm} = \text{TRUE}$. which means remove the NA values.

Example:

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help, and Addins. The top right corner shows 'Project: (None)'. The left pane contains a script editor with a file named 'temp.R' containing R code to calculate the mean of a vector. The right pane shows the Environment viewer with the results of the code execution. The bottom pane shows the Console output.

```

temp.R
1 # Create a vector.
2 x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
3
4 # Find mean.
5 result.mean <- mean(x)
6 print(result.mean)
7
8 # Find mean dropping NA values.
9 result.mean <- mean(x,na.rm = TRUE)
10 print(result.mean)

10:19 | (Top Level) | R Script | R 4.1.2 · ~/R
> # Create a vector.
> x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
>
> # Find mean.
> result.mean <- mean(x)
> print(result.mean)
[1] NA
>
> # Find mean dropping NA values.
> result.mean <- mean(x,na.rm = TRUE)
> print(result.mean)
[1] 8.22
>

```

d. Least Square Variables:

⑨Least square regression is a technique that helps us draw a line of best fit depending on the data points. The line is called the least square regression line., which perfectly depicts the changes in y(dependent variables) and their corresponding x(independent variables). This line is used to predict the value of y for particular value of x. Having both the dependent variables and independent variables is the first requirement of any regression technique.

⑩The least-squares method is a crucial statistical method that is practised to find a regression line or a best-fit line for the given pattern. This method is described by an equation with specific parameters. The method of least squares is generously used in evaluation and regression. In regression analysis, this method is said to be a standard approach for the approximation of sets of

equations having more equations than the number of unknowns.

The method of least squares actually defines the solution for the minimization of the sum of squares of deviations or the errors in the result of each equation.

Let us assume that the given points of data are $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ in which all x 's are independent variables, while all y 's are dependent ones. Also, suppose that $f(x)$ be the fitting curve and d represents error or deviation from each given point.

The least-squares explain that the curve that best fits is represented by the property that the sum of squares of all the deviations from given values must be minimum.

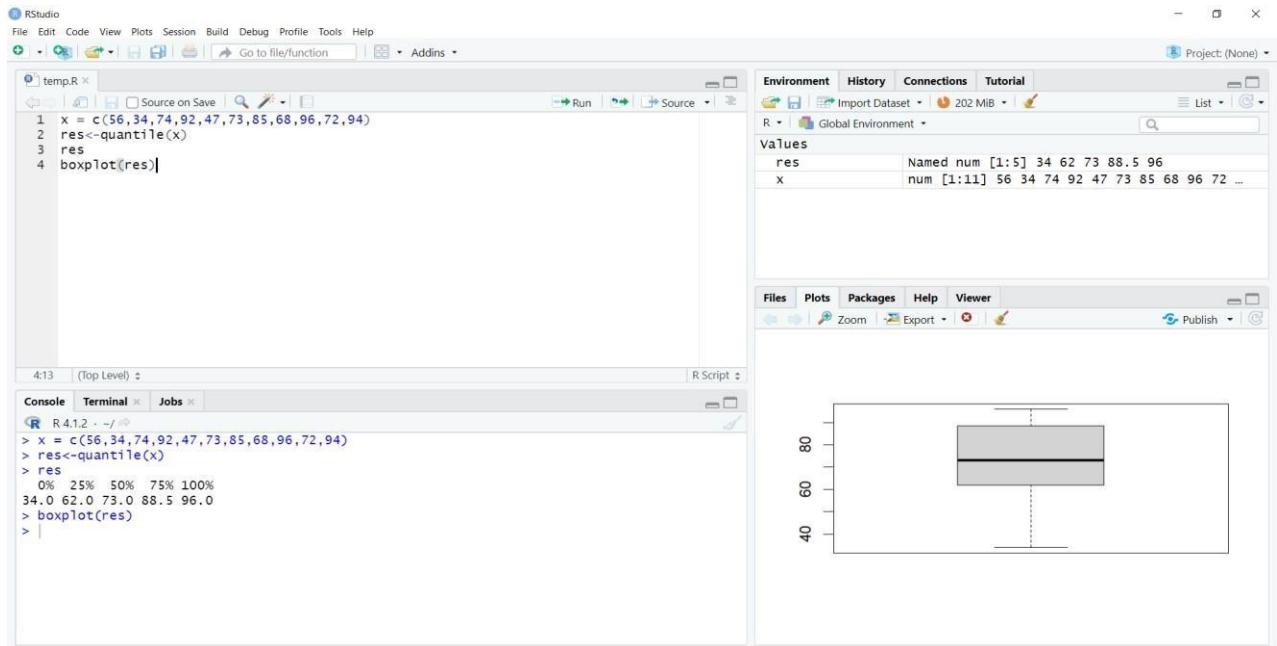
- ⑨ The premise of a regression model is to examine the impact of one or more independent variables (in this case time spent writing an essay) on a dependent variable of interest (in this case essay grades). Linear regression analyses such as these are based on a simple equation: $\text{Y} = a + b\text{X}$

Here Y is dependent variable and X is independent variable.

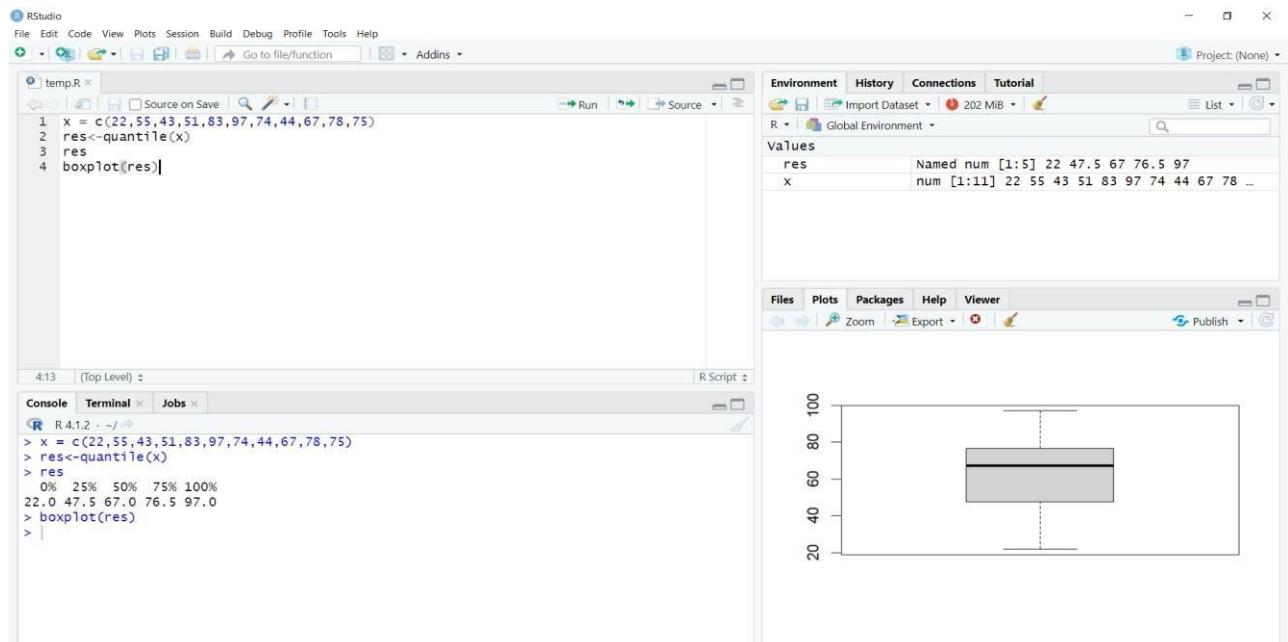
LAB ASSIGNMENT-A7

1. Calculate and visualize the Quartiles of the following set of data using R programming.

a. 56,34,74,92,47,73,85,68,96,72,94



b. 22,55,43,51,83,97,74,44,67,78,75



2. How to calculate the percentiles of set of student marks at a given probabilities and visualize them through bar-chart using R programming.

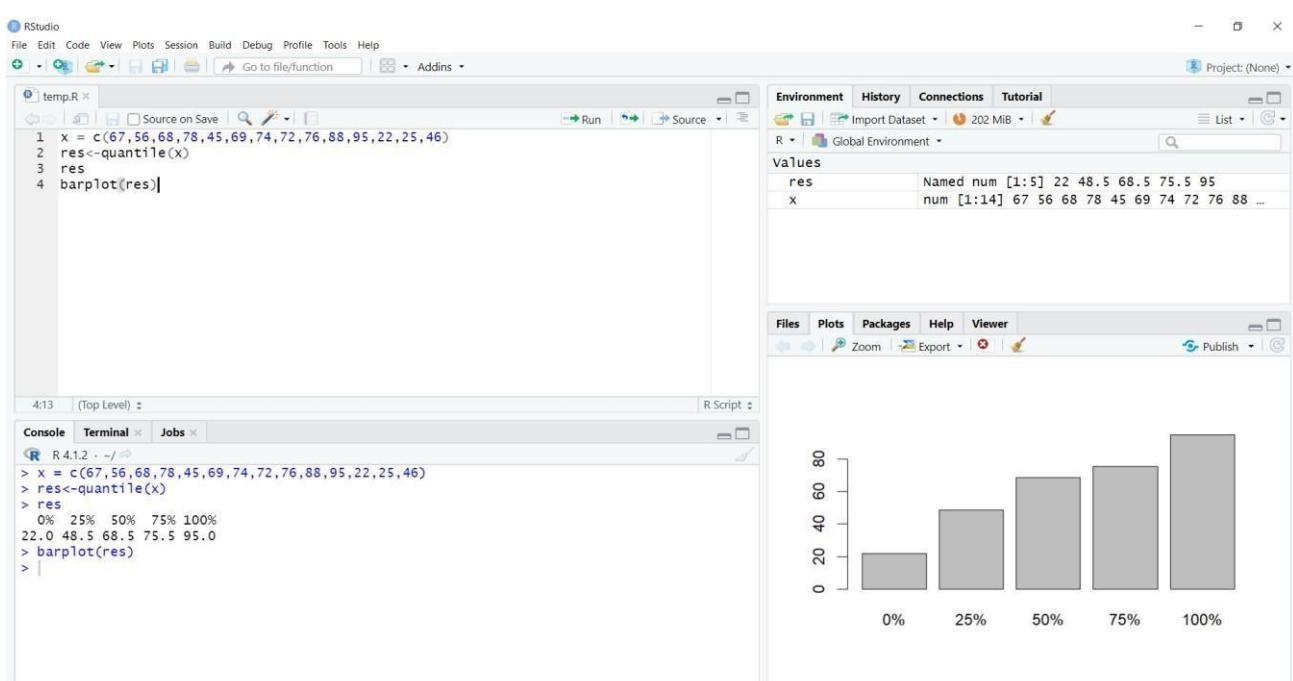
First, we need to store the marks of the students in a vector to combine the list of marks of the students. To calculate the percentiles of set of student marks at a given probabilities , we use quantile function i.e.,

`quantile()` -the quantile function, associated with a probability distribution of a random variable, specifies the value of the random variable such that the probability of the variable being less than or equal to that value equals the given probability.

Now, we need to store the quantile so calculated in a variable. To visualize the percentiles of the set of student marks so obtained using the quantile function, we should use the `barplot()` function.

R uses the function `barplot()` to create bar charts

a. 67,56,68,78,45,69,74,72,76,88,95,22,25,46.



b. 34,78,88,89,56,67,65,54,64,73,84,86,82,61

